N° d'ordre : 3206

THESE

En vue de l'obtention du : **DOCTORAT**

Structure de Recherche : Laboratoire Informatique, Mathématique Appliquée, Intelligence Artificielle et Reconnaissance de Formes
Discipline : Sciences de l'ingénieur
Spécialité : Informatique et Intelligence artificielle

Présentée et soutenue le : 21/05/2019 par :

Nabila ZRIRA

Contribution to the development of algorithms based Deep Learning architectures for mobile robotic's applications

JURY

Ahmed TAMTAOUI	*PES, Institut National des Postes et Télécommunications, Rabat*	*Président*
El Houssine BOUYAKHF	*PES, Université Mohammed V, Faculté des Sciences, Rabat*	*Directeur de thèse*
Mohammed Majid HIMMI	*PES, Université Mohammed V, Faculté des Sciences, Rabat*	*Codirecteur de thèse*
Fatima Zahra BELOUADHA	*PES, Université Mohammed V, Ecole Mohammadia d'ingénieurs, Rabat*	*Examinatrice*
Fouzia OMARY	*PES, Université Mohammed V, Faculté des Sciences, Rabat*	*Rapportrice/Examinatrice*
Ibtissam BENMILOUD	*PES, Ecole Nationale Supérieure des Mines, Rabat*	*Rapportrice/Examinatrice*
Mustapha METAICH	*DR, Elandaloussi Metaich Consultants, Casablanca*	*Invité*

Année Universitaire : 2018/2019

Acknowledgements

This research doctoral project was carried out at LIMIARF Laboratory (Laboratoire d'Informatique Mathématiques Intelligence Artificielle et Reconnaissance de Formes) at Mohammed V University, Faculty of Sciences, Rabat, Morocco, during the years 2014-2019 under the supervision of Prof. El Houssine BOUYAKHF and Prof. Mohammed Majid HIMMI.

This dissertation would not have been possible without the support of many people and therefore it is almost unfeasible to thank everyone. In many case, I do not even have the words to express all my gratitude.

First and foremost, I would like to express my thanks to Prof. El Houssine BOUYAKHF (PES, University Mohammed V, Faculty of Sciences, Rabat), my supervisor, without whom I would never have the chance to become a member of this one of the top computer science laboratories at Faculty of Sciences Rabat. He deserves my deepest gratitude for taking me into the group, for providing excellent facilities, and the wonderful inspiring atmosphere. Furthermore, for his many suggestions, constant support during this research, and highly appreciated experience which led to many happy memories. Without his patience, understanding, and wisdom, I would not be able to finish my Ph.D. degree.

I would also like to warmly thank Prof. Mohammed Majid HIMMI (PES, University Mohammed V, Faculty of Sciences, Rabat) for the help, the insights and the comments he gave me without restraint. I am glad he accepted me to be a member of the LIMIARF laboratory.

I would like to extend my gratitude to Prof. Ahmed TAMTAOUI (PES, National Institute of Posts and Telecommunications, Rabat) for honouring my jury by presiding it.

I am most grateful to my reporter Prof. Fouzia OMARY (PES, University Mohammed V, Faculty of Sciences, Rabat) for the precious comments.

I would like to thank Prof. Ibtissam BENMILOUD (PES, National School of Mines, Rabat) for the examination of the dissertation. Thanks a lot for being a jury and being part of my thesis defense.

In the same way, I wish to thank Prof. Fatima Zahra BELOUADHA (PES, University Mohammed V, Mohammadia School of Engineering, Rabat) for accepting to be a member of the jury of this thesis.

I would like to express my thanks to Dr. Mustapha METAICH (GM, Elandaloussi Metaich Consultats, Casablanca) for accepting to be a member of the jury of this thesis.

Special thanks are due to Prof. Benabdallah, Prof. Belhamidi, and Prof. Bahmad who helped me a lot in the difficult period. I thank them for their help, for their encouragements, and their supports.

I would like to thank my coauthors Haris Ahmad Khan, João Paulo Virgílio Marinho Martins, Saquib Ahmad. I enjoyed friendly working, discussing and collaborating with you. This was and will always be a pleasure to work with you.

Thanks to my dear friends Soufiana Mekouar, Ghizlane El Khettabi, Fatima Zahra Ouadiay, Samia Benchekroun, Hanae Ifzarne, Rihab Boussetta, Mohamed Hannat, El Mehdi El Graoui, Salma Tayeb, Issam ELAFI, Hajar Namrane, Rajae Haouari, and Mustapha Hammouny for the time spent to enjoy lunch and tea talks.

Thanks to all my friends who I may forget to notice by name for supporting me, to have fun and to enjoy my life of happiness and joy.

Most importantly, my deepest gratitude and special thanks go to my dear parents Lalla Fatiha Smaili and Mohammed Zrira who have always filled my life with joy, endless encouragement, generous love, and unconditional support. This academic degree will be truly impossible without their support and encouragement. My father is no more with us but I am sure that his soul will be very happy and proud of all that I have achieved. May his soul rest in peace.

My thanks go also to my sister Mariem Zrira, my nephews Ijlal and Mohammed Jalal, and my brother Hicham Zrira. Their love gave me forces to make this work.

My warm and special thank go to my uncles Fattah Douk and Abdelhak Zrira as well as all my family for their endless moral support throughout my life. To them, I dedicate this thesis.

Nabila ZRIRA

Abstract

This dissertation presents a direct application of deep learning architectures in different mobile robotic tasks. It is encompassed in three major parts. In the object classification part, we propose several approaches using 2D/3D descriptors and Deep Belief Networks (DBNs). First, we evaluate the most existing Point Cloud Library's (PCL's) descriptors by proposing a new recognition pipeline of 3D point clouds. Second, we propose many local and global approaches for classifying both 2D and 3D objects using 2D/3D Bag of Words (BOWs) as well as our new global descriptor Viewpoint Features Histogram-Color (VFH-Color). Third, a global approach for representing and learning 3D object categories using a global descriptor and DBN architectures is proposed.

The second part of this dissertation tackles the scene classification including two main contributions. The first one is centered on biologically inspired methods for representation and classification of indoor environments. The second contribution provides a new multimodal feature fusion for robust RGBD indoor scene classification.

The last part presents our contributions in topological navigation field. First, we propose a new method of exploring indoor environments by an autonomous mobile robot, as well as building topological maps. Second, we extend our previous work of topological navigation by using Convolution Long Short-Term Memory (C-LSTM) in order to perform scene recognition-based topological mapping and localization.

Keywords: Mobile robotic, deep learning, Point Cloud Library, object classification, 3D descriptors, VFH-Color, scene classification, GIST features, topological navigation, Deep Belief Network, Convolutional Neural Network, and Long Short-Term Memory.

Résumé

Cette thèse présente une application directe des architectures d'apprentissage profond dans différentes tâches de la robotique. Elle se subdivise en trois parties principales. Dans la partie classification d'objets, nous proposons plusieurs approches qui utilisent des descripteurs 2D/3D ainsi que des réseaux de croyance profonds. Dans un premier temps, nous évaluons les descripteurs les plus existants de la bibliothèque de nuages de points en proposant un nouveau pipeline de la reconnaissance des nuages de points 3D. Deuxièmement, nous proposons de nombreuses approches locales et globales pour classer les objets 2D et 3D à l'aide du sac de mots 2D/3D ainsi que notre nouveau descripteur global Viewpoint Features Histogram-Color (VFH-Color). Troisièmement, une approche globale pour représenter et apprendre des catégories d'objets 3D à l'aide d'un descripteur global et des réseaux de croyance profonds est proposée.

La deuxième partie aborde la classification des scènes qui comprend deux contributions principales. La première est centrée sur des méthodes biologiquement inspirées pour la représentation et la classification des environnements intérieurs. La deuxième contribution fournit une nouvelle fusion multimodale de caractéristiques pour une classification robuste des scènes intérieures RGBD.

La dernière partie de cette thèse présente nos contributions dans le domaine de la navigation topologique. Nous proposons tout d'abord une nouvelle méthode d'exploration d'un environnement intérieur par un robot mobile autonome, ainsi que la création de cartes topologiques. Deuxièmement, nous étendons notre travail précédent de la navigation topologique en utilisant la convolution et la mémoire à long-court termes (C-LSTM) afin de réaliser la cartographie et la localisation topologiques basées sur la reconnaissance des scènes.

Mots clés: Robotique mobile, apprentissage profond, nuages de points, classification d'objets, descripteurs 3D, VFH-Color, classification des scènes, caractéristiques GIST, navigation topologique, réseau de croyances profond, réseau de neurones à convolution, et mémoire à long-court termes.

Résumé étendu

Le premier essor de la robotique est accordé au développement des robots industriels. Ces derniers sont des machines capables de manipuler des objets et de réaliser des tâches d'une manière automatique selon un programme bien précis. Ils sont généralement utilisés afin de suppléer les humains pour des tâches dangereuses, fatigantes, de haute précision ou répétitives. Ces robots agissent dans des environnements parfaitement contrôlés en exécutant des tâches entièrement prédictibles et définies a priori. De nos jours, grâce aux progrès technologique et scientifique, les robots ont révolutionné non seulement le domaine industriel mais ils sont envisagés pour des tâches plus complexes comme l'assistance aux personnes âgées ou à mobilité réduite ainsi que la conduite autonome. Ces robots autonomes confrontent une diversité d'environnements dynamiques et rencontrent de nombreux défis. Considérons l'exemple d'un robot domestique d'assistance personnelle, un tel robot doit effectuer certaines tâches telles que trier et plier les vêtements, ramasser et nettoyer, faire fonctionner des appareils (une télévision, une machine à laver, etc.), ou préparer des aliments dans la cuisine. De plus, il doit gérer la grande variété d'objets, de meubles et de matériaux associés à ces tâches qui, dans certains cas, n'ont jamais été vus auparavant par le robot, ainsi que les endroits où ils sont déposés. Traditionnellement, un roboticien concevait manuellement les contrôleurs pour chaque tâche que les utilisateurs humains peuvent effectuer d'une manière intuitive. Ces contrôleurs peuvent être très complexes à concevoir car il s'avère extrêmement difficile de traduire cette intuition naturelle en un ensemble de fonctions et programmes. En outre, il s'avère difficile d'adapter ces approches à la grande variété des solutions auxquelles le robot doit faire face dans son environnement dynamique. Pour ces raisons, il est intéressant de doter les robots de mécanismes d'apprentissage automatique leur donnant ainsi la possibilité de construire eux-mêmes des représentations adaptées à leur environnement. Bien que les algorithmes d'apprentissage automatique présentent de nombreux avantages pour les applications en robotique mobile, ils peuvent néanmoins être difficiles à les appliquer à de nouveaux problèmes. D'une part, ces algorithmes nécessitent une optimisation puisqu'ils prennent énormément de temps pour effectuer des inférences, ce qui les rend impossibles pour des applications robotiques soumises à des contraintes de temps strictes. D'autre part, la conception des modèles qui sont assez généraux pour s'appliquer à d'autres cas du même problème, tout en restant suffisamment spécifiques pour s'y adapter, peut être très difficile, en particulier avec l'énorme diversité de la robotique dans le monde réel. Plus récemment, les approches d'apprentissage profond (deep learning) ont montré des performances impressionnantes dans un large éventail de domaines, y compris la vision par ordinateur, le traitement audio, le traitement du langage naturel, etc. Ces algorithmes sont basés sur les réseaux de neurones, des modèles hautement paramétrés qui utilisent plusieurs couches de représentation pour transformer des données en une représentation spécifique à une tâche. Ces réseaux profonds sont des modèles non-linéaires avec un nombre extrêmement élevé de paramètres (généralement de l'ordre de plusieurs millions), ils constituent en réalité des modèles non-linéaires généraux, capables d'apprendre toute mise en correspondance fonctionnelle des entrées aux sorties. Ceci est utile pour les applications

robotiques, qui rencontrent généralement une vaste gamme de non-linéarité, dont beaucoup sont difficiles ou impossibles à modéliser. Notre sujet de thèse s'inscrit principalement dans les domaines de la navigation, la perception, et l'apprentissage pour la robotique mobile. Ce sujet orienté vers une robotique «cognitive», a pour objectif général de permettre au robot de s'adapter à son environnement. Ce dernier est typiquement un milieu intérieur (appartement, laboratoire, etc.), donc un ensemble de pièces meublées, contenant de nombreux objets, supposés statiques. Ces objets peuvent être fixes (étagères, tableaux, etc.) ou peuvent être déplacés par l'homme (sirop, bouteille, pomme, etc.). Tout d'abord, le robot doit identifier tous les objets ainsi que les lieux de ce milieu en utilisant uniquement les informations visuelles. Par la suite, ces caractéristiques seront apprises par des méthodes d'apprentissage profond afin de construire des modèles appropriés pour chaque application. Après l'identification des objets et des lieux (scènes), le robot doit être en mesure de cartographier et de se localiser dans cet environnement pour assurer sa navigation. Avant de présenter nos contributions, il était crucial d'introduire dans Chapitre 2 les définitions, les motivations et les architectures de l'apprentissage profond. Tout d'abord, nous avons défini la forme de base la plus fondamentale des réseaux de neurones artificiels en fournissant l'analogie entre un neurone artificiel et un neurone biologique. Ensuite, nous avons introduit les architectures d'apprentissage profond, à savoir les machines de Boltzmann restreintes (restricted Boltzmann machines) et ses variantes, les réseaux de croyance profonds (Deep Belief Network-DBN), les réseaux de neurones à convolution (Convolution Neural Network-CNN), les réseaux de neurones récurrents (Recurrent Neural Network-RNN) et les réseaux récurrents à mémoire court et long termes (Long Short-Term Memory-LSTM) ainsi que leur processus d'apprentissage. Ce chapitre est une étude exhaustive des différents réseaux de neurones profonds qui ont été adaptés pour les applications de robotique dans les chapitres qui suivent. Les contributions de cette thèse se subdivisent en trois parties principales: la classification d'objets, la classification des scènes, et finalement la navigation topologique. Etant donné que ces parties sont différentes, nous avons débuté chacune par un chapitre qui englobe la plupart des travaux connexes du domaine étudié.

Part I : Classification d'objets

La reconnaissance d'objets est une tâche fondamentale pour un grand nombre d'applications de la robotique mobile telles que la saisie d'objets, la reconnaissance des scènes, ou la cartographie et la localisation simultanées (Simultaneous Localization and Mapping-SLAM). Dans cette partie, nous avons proposé plusieurs contributions qui se basent essentiellement sur l'extraction des points d'intérêts et des descripteurs 3D à partir des nuages de points 3D, puis l'apprentissage de ces derniers en utilisant différents types de réseaux de croyance profonds.

Chapitre 3 Classification d'objets : Etat de l'art

Dans ce chapitre, nous avons fourni dans un premier lieu une brève discussion sur les travaux connexes de la reconnaissance d'objets qui sont catégorisés en des approches de reconnaissance et de catégorisation 2D, des approches de reconnaissance et de catégorisation 3D, puis des approches basées sur l'apprentissage profond. Dans un deuxième lieu, nous avons introduit notre contribution dans le contexte de la reconnaissance d'objets 3D. Afin de construire notre propre base de données d'objets 3D, nous avons utilisé le module de l'acquisition des nuages de points 3D du logiciel RGBDemo. Ensuite, nous avons évalué les méthodes d'extraction de caractéristiques existantes issues de la bibliothèque publique PCL (Point Cloud Library). En outre, nous avons suggéré un nouveau pipeline de reconnaissance de nuages de points 3D basé sur les descripteurs PCL et un seuil de rejet qui détermine la bonne correspondance entre l'objet du test et ceux qui existent dans la base de données d'apprentissage. Les principales contributions sont:

- l'acquisition d'un modèle d'objet 3D à l'aide du logiciel RGBDemo et le support rotatif;
- l'extraction des objets qui sont présents dans les scènes 3D;
- l'évaluation des descripteurs 3D les plus existants de la version 1.7 de PCL;
- le calcul du seuil de rejet pour chaque classe d'objets (nuage de points) afin d'éviter les mauvaises classifications et d'améliorer le taux de classification.

Chapitre 4 Contributions à la reconnaissance et à la catégorisation d'objets 2D/3D

Dans ce chapitre, nous avons proposé plusieurs approches locales et globales pour la classification d'objets 2D et 3D en utilisant diverses informations. Pour cela, nous avons décrit une base de données d'objets 2D et des nuages de points 3D avec des descripteurs locaux 2D/3D que nous avons quantifié par la suite avec l'algorithme k-means pour obtenir le sac de mots (Bag of Words-BoWs). De plus, nous avons développé un nouveau descripteur global nommé «VFH-Color» qui combine la version originale du descripteur histogramme de point de vues (Viewpoint Feature Histogram-VFH) avec l'histogramme de quantification de couleur, ajoutant ainsi les informations d'apparence qui améliorent le taux de la reconnaissance. Ensuite, nous avons appris séparément ces descripteurs en utilisant le réseau de croyance profond. En résumé, nos principales contributions sont:

- la description d'une base de données des images 2D avec les caractéristiques SURF (Speeded Up Robust Features) qui sont quantifiées par l'algorithme k-means pour créer le sac de mots 2D;
- la description des nuages de points 3D avec les caractéristiques SI (Spin Images) qui sont quantifiés par l'algorithme de groupement k-means afin de générer le sac de mots 3D;
- la proposition du nouveau descripteur «VFH-Color» combinant les informations de couleur et les caractéristiques géométriques extraites de la version précédente du descripteur VFH;
- l'apprentissage des différentes caractéristiques en utilisant le réseau de croyance profond et la comparaison des résultats avec les machines à vecteurs de support.

Chapitre 5 Réseaux de croyance profonds génératif et discriminant pour la catégorisation d'objets 3D

Ce chapitre a proposé une nouvelle approche globale pour la représentation et l'apprentissage des catégories d'objets 3D à l'aide des descripteurs globaux et des architectures d'apprentissage profond. Comme les descripteurs globaux décrivent l'objet en entier, une étape de prétraitement est généralement nécessaire pour supprimer les plans et les murs de la scène 3D, puis la segmenter en différents objets qui la constituent. Après cette étape de segmentation, nous avons extrait les caractéristiques géométriques des nuages de points 3D à l'aide du descripteur VFH, puis nous avons appris les modèles de ces caractéristiques en utilisant les réseaux de croyance profonds. Par la suite, nous avons évalué la performance des réseaux de croyance profonds génératif et discriminant. Le réseau de croyance profond génératif entraîne une séquence des machines de Boltzmann restreintes tandis que le discriminant utilise une nouvelle architecture profonde basée sur les machines de Boltzmann restreintes et le modèle mixte de densité. Différentes techniques d'apprentissage des machines de Boltzmann restreintes ont été également évaluées, notamment la divergence contrastive (contrastive divergence), la divergence contrastive persistante (persistent contrastive divergence) et l'énergie libre dans la divergence contrastive persistante (free energy in persistent contrastive divergence). Les principales contributions de ce chapitre sont:

- la proposition d'un nouveau pipeline de catégorisation d'objets 3D basé sur un descripteur VFH et des architectures d'apprentissage profond;
- la segmentation de tous les objets présents dans les scènes 3D;
- l'extraction des caractéristiques géométriques à l'aide du descripteur VFH;
- l'apprentissage des caractéristiques extraites avec les réseaux de croyance profonds génératif et discriminant afin de montrer la différence entre l'apprentissage génératif et discriminant pour la catégorisation d'objets 3D.

Part II : Classification des scènes

Le problème de la classification des scènes est l'un des défis les plus difficiles en vision par ordinateur et robotique. Dans cette partie, nous avons proposé deux contributions qui ont montré des résultats très important par rapport à l'état de l'art de la classification des scènes 2D et 3D. La première contribution est une approche bio-inspirée qui se base sur l'extraction des caractéristiques issues de l'attention visuelle et l'apprentissage par les réseaux de croyance profonds. Par contre, dans la deuxième contribution l'extraction des caractéristiques et l'apprentissage de ces derniers sont réalisés en utilisant les réseaux de neurones à convolution.

Chapitre 6 Classification des scènes : Etat de l'art

Dans ce chapitre, certains des concepts importants de l'attention visuelle sont introduits d'un point de vue neurobiologique psychophysique. Ensuite, les approches de classification des scènes sont illustrées et catégorisées en trois familles: les approches basées sur l'attention visuelle, les approches de reconnaissance et de catégorisation 2D, et finalement les approches de reconnaissance et de catégorisation 3D.

Chapitre 7 Réseau de croyance profond discriminant pour la classification des environnements intérieurs en utilisant des caractéristiques visuelles globales

Dans ce chapitre, nous avons suggéré une nouvelle approche centrée sur des méthodes inspirées de la biologie pour la représentation et la classification des environnements intérieurs. Tout d'abord, les caractéristiques visuelles globales sont extraites à l'aide du descripteur GIST, puis nous avons utilisé ces dernières pour l'apprentissage du réseau de croyance profond. Ce réseau utilise une nouvelle architecture profonde basée sur les machines de Boltzmann restreintes et le modèle mixte de densité. Par la suite, la technique de rétro-propagation (backpropagation) est utilisée sur l'ensemble du réseau pour affiner les poids en vue d'une classification optimale. Nos principales contributions sont les suivantes:

- le développement d'un système de classification des environnements intérieurs en utilisant des méthodes inspirées de la biologie et basées sur les caractéristiques de GIST et des architectures profondes;
- l'extraction des caractéristiques visuelles globales à partir des bases de données réelles et synthétiques à l'aide du descripteur GIST;
- l'utilisation des différentes méthodes d'apprentissage des machines de Boltzmann restreintes: divergence contrastive, divergence contrastive persistante et énergie libre dans la divergence contrastive persistante;
- l'utilisation de la stratégie de rétro-propagation pour optimiser les résultats de la classification;
- la comparaison de notre approche avec les réseaux de neurones à convolution.

Chapitre 8 Fusion multimodale des caractéristiques pour la classification robuste des scènes intérieures RGBD

Dans ce chapitre, nous avons proposé une nouvelle fusion multimodale des caractéristiques pour la classification robuste des scènes intérieures RGBD. Notre architecture se compose de deux réseaux de neurones à convolution distincts appris sur les images RGB et les images de profondeur, puis combinés avec un réseau de fusion. De ce fait, nous avons introduit une méthode simple de coloration des images de profondeur afin de les utiliser comme entrée des réseaux de neurones à convolution. Ensuite, nous avons appris les images RGB ainsi que les images de profondeur colorées séparément à l'aide des réseaux de neurones à convolution qui sont pré-appris sur la base de données des scènes «Places», suivi d'une troisième étape d'apprentissage dans laquelle l'architecture affine les deux modalités avec un réseau de fusion qui effectue la classification finale. En résumé, les principales contributions sont:

- la suggestion d'une fusion multimodale des caractéristiques pour la classification des scènes RGBD;
- la proposition d'une nouvelle méthode simple de coloration des images en profondeur basée sur la mise à l'échelle polynomiale pour augmenter la précision de la classification;
- la combinaison des deux modalités des images RGB et des images de profondeur colorées avec un réseau de fusion qui effectue la classification finale.

Partie III : Navigation topologique

La navigation, plus particulièrement la cartographie, est une tâche primordiale pour les robots mobiles autonomes. Dans cette partie, nous avons élaboré un nouveau concept de la cartographie topologique en temps réel. La première contribution est basée sur l'apparence et comprend l'extraction des descripteurs globaux GIST à partir des images omnidirectionnelles. Pour la deuxième contribution, nous avons utilisé la reconnaissance des lieux dans le but de cartographier l'environnement du robot.

Chapitre 9 Navigation topologique : Etat de l'art

Ce chapitre est une définition concise de la navigation, ses stratégies ainsi que les différentes cartes de l'environnement. Il comprend également l'état de l'art des approches de la navigation y compris les approches robustes à l'aliasing perceptuel, les approches probabilistes, les approches basées sur l'apparence, les approches basées sur la reconnaissance des lieux, les approches basées sur la mémoire et finalement les approches basées sur l'apprentissage profond.

Chapitre 10 Une nouvelle cartographie topologique incrémentale en utilisant des caractéristiques visuelles globales

Le but de ce chapitre est de présenter une nouvelle méthode d'exploration des environnements intérieurs par un robot mobile autonome, ainsi que la construction des cartes topologiques basées sur des informations visuelles globales. Dans ce chapitre, nous avons utilisé également des images omnidirectionnelles pour construire un seul descripteur visuel global qui décrit la scène en entier. De plus, afin de résoudre le problème de la fermeture de boucle visuelle, nous avons proposé une formule qui attribue correctement chaque descripteur global à son emplacement. Notre contribution dans ce chapitre permet de construire la carte topologique incrémentale à partir des images omnidirectionnelles et s'appuie sur les critères suivants:

- l'assurance d'une cartographie topologique incrémentale de l'environnement;

- l'utilisation des descripteurs globaux de petite taille "GIST" pour le calcul de la signature de chaque scène;
- le calcul du seuil pour la détection de la fermeture de boucle.

Chapitre 11 Apprentissage de bout en bout pour naviguer dans un environnement intérieur

Récemment, la reconnaissance des lieux est devenue une problématique importante pour les robots mobiles autonomes. Elle est principalement utilisée pour la navigation de robots dans des environnements intérieurs. Dans ce chapitre, nous avons proposé une nouvelle approche de navigation topologique basée sur la reconnaissance des lieux. Pour que le robot puisse explorer et naviguer dans son environnement, il doit être capable d'identifier les lieux de ce dernier. À cette fin, nous avons utilisé l'architecture C-LSTM (Convolutional Long Short-Term Memory) pour apprendre les scènes lors de la navigation du robot mobile. C-LSTM implique des couches de réseau de neurones à convolution (CNN) pour extraire les caractéristiques des données d'entrée combinées à une mémoire à court long termes (LSTM) afin de prendre en compte les informations des séquences précédentes et d'apprendre les dépendances temporelles du mouvement du robot. Pour l'extraction des caractéristiques, nous avons utilisé le CNN pré-appris sur la base de données des scènes «Places», par la suite ces caractéristiques extraites sont utilisées comme entrée du LSTM. Après avoir effectué la reconnaissance des scènes, le robot crée et mis à jour sa carte topologique afin d'agir dans son environnement. En résumé, les principales contributions sont:

- la proposition d'une nouvelle approche basée sur la reconnaissance des lieux pour la navigation topologique;
- l'utilisation de l'architecture C-LSTM pour apprendre les lieux de l'environnement;
- l'utilisation du CNN pré-appris sur «Places» pour extraire les caractéristiques et LSTM pour apprendre les dépendances temporelles des données;
- la construction d'une carte topologique basée sur la reconnaissance des scènes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Most frequently asked question at my AI talks: Will robots be conscious?"

Piero Scaruffi

Contents

WHEN people ask what is my topic of research, and I reply "I'm working on mobile robotic applications such as object recognition, place or scene classification, and topological navigation using deep learning", I get one of these possible feedbacks:

- are robots so intelligent and autonomous to perform these tasks?
- can a robot bring, for example, a coffee mug from the kitchen?
- can a robot navigate in its environment by identifying the visited places?
- what are the reasons for the resurgence of deep learning? and how it can be used effectively in mobile robotic applications?

In recent years, robots are deployed in many areas where automation and decision-making skills are required. Robots are not just mechanically advanced but are becoming intelligent as well. The idea behind these intelligent machines is the creation of systems that imitate the human behavior to be able to perform tasks which are actually infeasible for humans. The type of tasks for which robots are well adapted includes those that are in an unexplored environment such as outer-space and undersea. However, the robot tasks are not limited just to complex and difficult problems, but they are covering some industrial, medical, and domestic applications. To accomplish such applications, mobile robots must acquire certain autonomy. The autonomy is an ability of an unmanned system to capture, perceive, analyze, communicate, plan, make decisions and act to achieve the goals, assigned to it by a human operator using a human-machine interface. The autonomy is spread over several levels, which are characterized by factors including the complexity of the mission, the environmental difficulties and the necessary level of human-robot interaction for the accomplishment of the mission. Through this faculty, mobile robots are able to adapt or make a decision in order to perform a task, despite a lack of preliminary or possibly erroneous information. Furthermore, the autonomy is relative to the mission objective taking into account all the capabilities of the robot, especially the perception one.

The human can search and find an object visually in a cluttered scene. It is a very simple task for a human to pick an object and place it in the required place while avoiding obstacles along the path, and without damaging the fragile objects. These simple and trivial tasks for humans become challenging and complex for robots and can overcome their capabilities. The pick-up and drop applications are performed in fully known and structured environments in which objects should be recognized by a domestic robot. The recognition systems are structured so that they first detect plane surfaces (e.g., a table or a desk) for restricting the search area of the possible object's positions, and then compute features in order to describe the objects. Popular features include the use of colors, edges, geometric forms or compute local interest points (i.e., keypoints) and descriptors (e.g., Scale-Invariant Feature Transform-SIFT and Speeded Up Robust Features-SURF). Another possible way is the use of 3D feature descriptors such as feature histograms obtained from range images or point clouds. Although object recognition can identify objects present in a scene. This remains unsuccessful in performing tasks for mobile robots. The domestic robot must be able to localize itself and navigate in its environment in order to determine a suitable and safe path between the starting point and the goal. This process is called *navigation*.

Human beings and most of the mammals do large-scale navigation by using topological maps and scene recognition. A topological map is a topological representation of the world where only places and connections between them are stored. Such representation is typically illustrated in form of a graph with nodes (i.e., places) and edges (i.e., connections). It provides an efficient means for autonomous robots to perform localization and path-planning. Humans use the vision system to navigate. This is a very complicated task for robots that only appears simple because humans do it so easily. In the last three decades, visual navigation has become a source of countless approaches since vision-based navigation can increase the scope of autonomous mobile robot applications.

Deep learning is typical bio-inspired methods when multiple processing layers are used to model high-level abstraction. Its origin backs to the Artificial Neural Network (ANN) paradigm, which is proposed in the 1940s by McCulloch and Pitts. Advancements in deep learning over the years in several research fields have attracted research into how deep artificial neural networks can be used in robotic systems. From the perspective of robotics, deep learning should solve the perception and decision-making problems, which have not been fully developed yet. However, deep learning has successfully solved several preliminary perception issues for robots such as object recognition and could solve other applications like robot motion, manipulation and grasping, automation, self-supervision, self-training, and learning.

1.1 Brief historical context of deep learning

Humans and animals can display behaviors that they label as intelligent by learning from experience. Learning offers them flexibility in their life; the fact that they can adjust and adapt to the new circumstances, and learn new tricks [150]. This ability to learn represents one of the most fundamental attributes of intelligent behavior that include the acquisition of new declarative knowledge, the development of cognitive skills through instruction or practice, the organization of new knowledge into general, effective representations, also the discovery of new facts and theories through observation and experimentation [161]. Ever since computers were invented, researchers have been interested in implanting such capabilities in computers. Imagine computers learning from medical records which treatments are most effective for new diseases, personal software assistants learning the evolving interests of their users in order to highlight especially relevant stories from the online morning newspaper, or houses learning from experience to optimize energy costs based on the particular usage patterns of their occupants [168]. The challenge to make computers learn nearly as well as humans learn has been, and remains, a most challenging and fascinating long-range goal in Artificial Intelligence (AI). The study and computer modeling of the learning process in their multiple manifestations constitute the subject matter of *machine learning*.

1.1.1 Hand-designed representations versus representation learning

Machine learning is inherently a multidisciplinary field which combines AI, computational complexity theory, probability and statistics, control theory, philosophy, psychology, information theory, neurobiology, and other fields [67]. The introduction of machine learning has allowed computers the ability to solve problems which involve knowledge of the real-world and make decisions that seem subjective. For instance, a simple algorithm called Naive Bayes can be used to filtering "junk" (i.e., often referred to as spam) e-mails on the Internet [9, 203]. The performance of this algorithm depends heavily on the representation of the given data. There are many particular data of e-mail message that provide evidence as to whether a message is legitimate or junk. Such words are known in machine learning as *features*. For example, particular sentences, such as "free money", or "you have won a vacation", are indicative features of junk e-mails. However, Naive Bayes cannot influence the way that the features are defined but it only learns how each of these features correlates with the text of a message in new e-mails.

Many AI tasks can be solved by designing a robust set of features for each specific task, then providing these features to a machine learning algorithm. However, the difficulty remains in deciding what features should be extracted? Suppose that we would like to classify the household objects in the kitchen. We know that Apple has a round shape, so we might like to use shapes as features. But at the same time tomato also has a similar shape to the apple. The problem becomes more complicated if we want to differentiate between these two objects using color information like features, especially that red apple can be confused with tomato. One solution to resolve this problem is to use machine learning to discover not only the mapping from representation to output but also the representation itself. This approach is called *representation learning*.

Learned representations of the data extract useful information when building classifiers or other predictors. They often result in much better performance than those obtained with hand-designed representations. They also allow AI systems to quickly adapt to new tasks, with minimal human intervention. Moreover, a representation learning can discover good features for a simple task in a few minutes, or a complex one in hours to months. Manually hand-designed features for a complex task requires a great deal of human time and effort; it can take years for an entire community of researchers [18]. The typical example of a representation learning algorithm is the *autoencoder*. An autoencoder is a simple learning circuit that aims to transform inputs into outputs with the least possible amount of distortion. It combines two functions:

1. encoder function which converts the input data into a different representation;
2. decoder function which converts the new representation back into the original format.

The autoencoder is trained to preserve as much information as possible when an input is run through the encoder function and then the decoder one. Also, it is trained to offer to the new representation diverse kind of properties which differ depending on the autoencoder type. The objective of designing algorithms for learning features is usually to separate the factors of variation that explain the observed data. The word *factors* is simply used to refer to separate sources of influence; they are usually not combined by multiplication. Such factors are often not quantities which are directly observed. Otherwise, they may exist either as unobserved objects or unobserved forces in the physical world that affect observable quantities. When analyzing a speech recording, the factors of variation include the subject's age, their sex, their accent and the words that they are speaking. When analyzing an image of a cat, the factors of variation include the position of the cat, its breed, its color, and the viewing angle. Many real-world AI applications encounter an inconvenient problem in the factors of variation which influence every single piece of data we are able to observe. The individual pixels in an image of a Bengal cat might be very close to the tiger. The fur of this cat can be very similar to the tiger fur. Instead, the shape of the cat's silhouette differs from the one of tiger and depends on the viewing angle. Most applications require us to disentangle the factors of variation and discard the ones that we do not care about. Certainly, it can be very difficult to extract such high-level, abstract features from raw data. Many of these factors of variation, such as animal breed, can be identified only using a sophisticated, nearly human-level understanding of the data. When it is nearly as difficult to obtain a representation as to solve the original problem, representation learning does not, at first glance, seem to help us.

Deep learning can solve this challenge in representation learning by introducing representations that are expressed depending on other, simple representations. It achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simple concepts, and more abstract representations computed in terms of less abstract ones.

Figure 1.1 illustrates an example of a deep learning model called feedforward deep network which contains one visible layer and three hidden layers. It represents just a mathematical function mapping input values to the output ones. The function is formed by composing many simple functions. We can think of each application of a different mathematical function as providing a new representation of the input. The visible layer is presented as the input pixels of the face image, which we are able to observe. Then a series of three hidden layers extract increasingly abstract features from the image. These layers are *hidden* in the sense that their values are not given in the data; instead, the model which must determine which concepts are significant for explaining the relationships in the observed data. Starting with the visible variables, the model first recognizes the most basic components; the first hidden layer can easily define edges by comparing the brightness of neighboring pixels. Then the second hidden layer can describe parts containing multiple components. Finally, the third layer can detect entire parts of the specific objects.

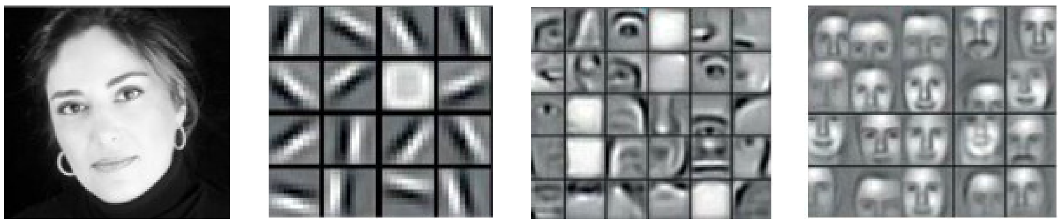

(a) Visible layer (b) Hidden layer 1 (c) Hidden layer 2 (d) Hidden layer 3

Figure 1.1: Visualization of a deep learning model. (a) Input pixels. (b) Edges at various orientations. (c) Object parts (i.e., combination of edges). (d) Object models.

1.1.2 Deep hierarchies in the primate visual cortex

Human processing mechanisms need deep architectures for extracting complex structure and building internal representation from rich sensory inputs. For example, the primate visual system is also characterized by its hierarchical and feedforward organization. The neuronal processing of visual information starts by the retina of the left and the right eyes. Nearly all connections are projected to a visual area named Lateral Geniculate Nucleus (LGN) before it reaches the visual cortex. We call these stages *precortical processing*. The visual cortex receives the direct inputs from the LGN and contains a very precise and orderly representation of the opposite half of the visual field. It is also organized into layers, where most of the feedforward connections (i.e., connections to a higher stage in the hierarchy) originate from the superficial layers and most of the feedback connections originate from the deeper layers [47, 78, 106, 146, 257]. The visual cortex hierarchy can be described by the following visual areas:

- visual area V1: V1 is the first cortical area which processes visual information. V1 neurons are most sensitive to low-level features than in LGN but remain relatively simple such as edges, gratings, line endings, motion, color, and disparity;

- visual area V2: V2 is a retinotopically-organized area that mostly receives its input from V1. The main new feature of V2 is the more sophisticated contour representation including texture-defined contours, illusory contours, and contours with border ownership;

- visual area V4: V4 neurons respond selectively to orientation, color, disparity, and simple shapes. They continue the process of integrating lower-level into higher-level responses and increasing invariances. New prominent features in V4 are curvature selectivity and luminance-invariant coding of hue;

- Inferior Temporal cortex (IT): many IT neurons, for example, are selective for the overall shape, color, or texture of a stimulus, anywhere with the central visual field. IT appears to be the last visual area in the cortical system for object recognition.

According to Figure 1.2, the information processing in a typical hierarchical feedforward model starts at the retina, proceeds to the LGN, then to V1, V2, V4, and IT. The lower visual areas have smaller receptive fields, while neurons in higher areas have gradually increasing receptive field sizes.

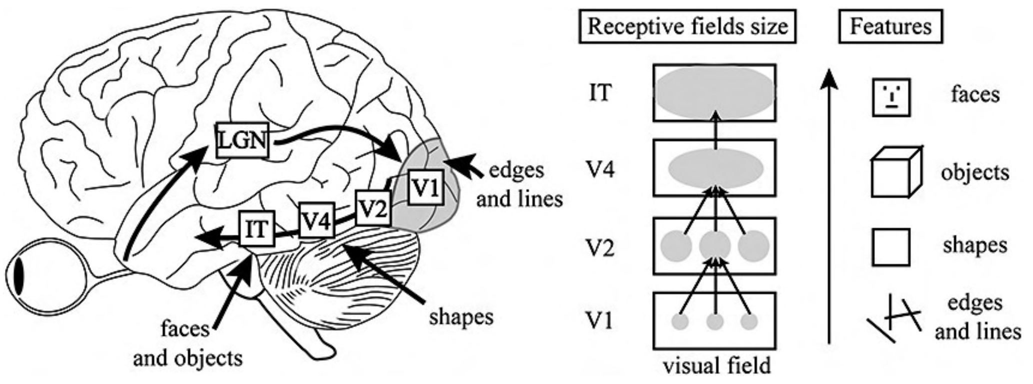
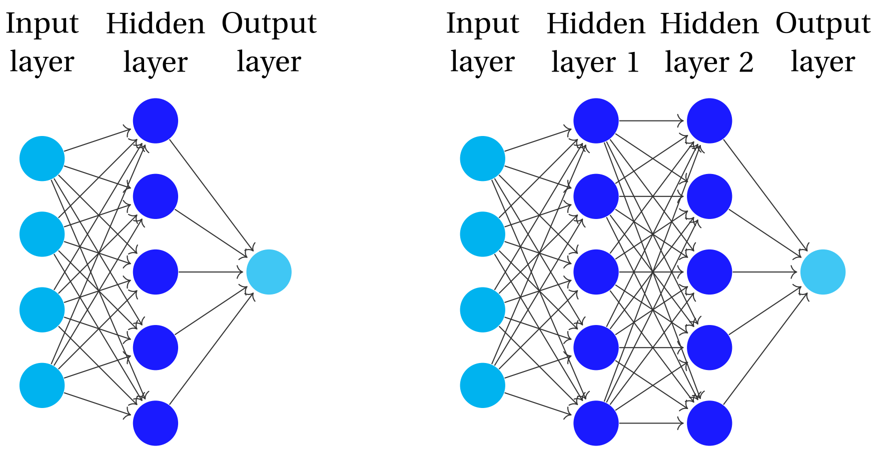


Figure 1.2: Left: feedforward model. Center: visual areas. Right: a hierarchy of features. The Figure is illustrated from [78].

1.1.3 From shallow to deep networks

Until recently, machine learning applications such as object detection, image classification, and signal processing had mostly exploited shallow architectures. Those are formed by a single layer of non-linear feature transformations and suffer from the absence of multiple layers of adaptive non-linear features [45]. The shallow architectures are conventional, generally used Hidden Markov Models (HMMs), Gaussian Mixture Models (GMMs), Support Vector Machines (SVMs), Conditional Random Fields (CRFs), Maximum Entropy (MaxEnt) models, logistic and kernel regression, and Multi-Layer Perceptron (MLP) neural network with a single hidden layer (Figure 1.3a). These shallow models share a common property; they are simple architectures that contain only one layer responsible for transforming the raw input signals or features into a problem-specific feature space, which may be unobservable.

Recent studies [19, 22] of non-parametric machine learning techniques particularly kernel methods, such as Support Vector Machine (SVM), semi-supervised learning algorithms, and graph-based manifold encounter some limitations in their ability to learn complex high-dimensional functions. The problem is obvious in kernel-based approaches when the kernel is "local" such as the Gaussian kernel. These studies mark the difficulty of learning "highly-varying functions"; functions that have a large number of "variations", i.e., they would need a large number of pieces to be well represented by a piecewise-linear approximation. The number of these pieces can be made to grow exponentially with the number of factors of variations in the input data. Hence, this growth causes the well-known curse of dimensionality for classical non-parametric learning algorithms (e.g., for classification, regression, and



(a) Shallow network (b) Deep network

Figure 1.3: (a) MLP with single hidden layer. (b) Deep network with two hidden layers.

density estimation). In case the shapes of all these pieces are unrelated, one needs enough examples for every piece to be generalized properly. However, if these shapes are related and can be predicted from each other, "non-local" learning algorithms have the potential to generalize to pieces not covered by the training set. Such an ability would seem necessary for learning in complex domains such as AI tasks.

Kernel machines -not only those with a local kernel- have a shallow architecture with only two layers [21]. The first one consists of fixed kernel functions, which match the incoming pattern with templates extracted from a training set. Whereas, the second layer is a linear combination of the matching scores. These architectures face a serious problem because they can be very inefficient in terms of the number of computational units such as hidden units, and thus in terms of required examples. One way to represent a highly-varying function compactly (i.e., with few parameters) is through the composition of many non-linearities, i.e., with a deep architecture. For instance, the parity function with d inputs needs $O(2^d)$ examples and parameters to be represented by a Gaussian Support Vector Machine (SVM), and $O(d^2)$ parameters for a simple neural network. In Figure 1.3b, we require $O(d)$ parameters and units for a multi-layer network with $O(log_2 d)$ layers. Unfortunately, when the representation of a concept requires an exponential number of elements, the number of training examples required to learn the concept may also be impractical.

In practice, shallow architectures have been shown successfully in solving many simple and well-constrained problems, but their limited modeling and representational power can cause difficulties when dealing with more complicated real-world applications involving natural signals such as human speech, natural sound as well as language, natural image, and visual scenes. Unlike the typical shallow architectures, the deep architecture can efficiently train a large number of parameters and therefore approximate high complexity functions, which are necessary for solving complex problems such as computer vision or natural language processing.

1.2 Motivation and research objectives

Our research focuses on mobile robotic applications using deep learning as well as information extracted from different types of sensors (i.e., RGB images, RGBD data, point clouds, and omnidirectional images). The aim of this thesis is to surpass traditional robot applications which are limited to only machine learning methods. Deep learning is introduced in the context of robotics as a means of making sense and analyzing the data extracted from different sensors with the use of multiple abstraction layers. With traditional robot technology, the data is extracted from sensors and set with old methods which are limited by constraints of adapting to generic settings. In situations where these robotic systems faced dynamic environments, they operated in an unstructured manner by combining hybrid and autonomous functionality to process information about their environment. Whereas, with the advent of deep learning, new methods of processing data from robotic sensors can automatically extract and learn features from the environment. For instance, in difficult environments such as a congested kitchen, the level of accurately perceiving the environment for mobile robots is limited. Therefore, deep learning-based solutions can tackle this challenge by processing deep layers to solve this complexity by successfully deciphering intricate environment perception difficulties. In summary, the main objectives of this thesis are:

- using deep learning methods in mobile robotic applications;
- proposing new 3D object classification approaches based on point clouds and our global descriptor called VFH-Color which combines the previous version of Viewpoint Feature Histogram (VFH) and color information;
- evaluating generative and discriminative Deep Belief Network architectures in the context of 3D object categorization in cluttered scenes captured from our laboratory;
- developing an indoor environment classification system by using biologically inspired methods based on GIST features and the power of Discriminative Deep Belief Network (DDBN);
- proposing a multimodal feature fusion for RGBD scene classification using Convolutional Neural Networks (CNNs) as well as colorized depth method;
- defining a new topological map concept of mobile robot environment;
- exploring indoor environments by an autonomous mobile robot and building topological maps based on global visual attributes extracted from omnidirectional images;
- proposing a new topological navigation approach based on Convolution Neural Network features and Long Short-Term Memory (LSTM).

1.3 Thesis organization

Figure 1.4 depicts a graphical roadmap summarizing the organization of the thesis. The contributions of this thesis consist of the reinvestigating of deep learning methods for mobile robotic applications: object classification, scene classification, and topological navigation. Chapter 2 presents a brief history and overview of neural networks that serve as a concise introduction to deep learning methods. Then it introduces the most used deep learning architectures such as Restricted Boltzmann Machine (RBM) and its variants, Deep Belief Networks (DBNs), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM). This chapter occupies a very important portion compared to the other chapters because it covers all the methods we have used through different parts of the thesis.

Divided into three distinct parts, the first part starts by presenting our contributions in object classification. Chapter 3 provides a brief discussion on the related work in 2D and 3D object classification, as well as the evaluation of Point Cloud Library's descriptors and our 3D recognition pipeline. Chapter 4 presents many local and global approaches for classifying both 2D and 3D objects using 2D/3D Bag of Words (BOWs) and our new global descriptor VFH-Color. The last Chapter of this part evaluates the Generative vs. Discriminative Deep Belief Network architectures for 3D object categorization on Washington RGBD dataset and our real-world scenes.

The second part of the thesis tackles the scene classification including two main contributions. In Chapter 6, some of the important concepts in visual attention, particularly bottom-up and top-down visual attention are introduced. Then, the existing state-of-art approaches to visual perception in computer vision is provided. Chapter 7 presents our proposed approach which is centered on biologically inspired methods for representation and classification of indoor environments. The approach combines GIST features and Discriminative Deep Belief Network (DDBN), which showed its performance in object classification. Chapter 8 provides a new multimodal feature fusion for robust RGBD indoor scene classification. Our last approach consists of two separate CNNs trained on RGB and depth images, then combined with a late fusion network.

Finally, the third part presents our contributions in topological navigation field. Chapter 9 illustrates the navigation strategies, the different maps used in mobile robotic navigation, and an exhaustive state-of-the-art navigation approaches. Chapter 10 presents our new method of exploring indoor environments by an autonomous mobile robot, as well as building topological maps based on global visual attributes which are extracted from omnidirectional images. Chapter 11 extends our previous work of topological navigation by using both CNN and LSTM in order to perform scene recognition-based topological mapping and localization.

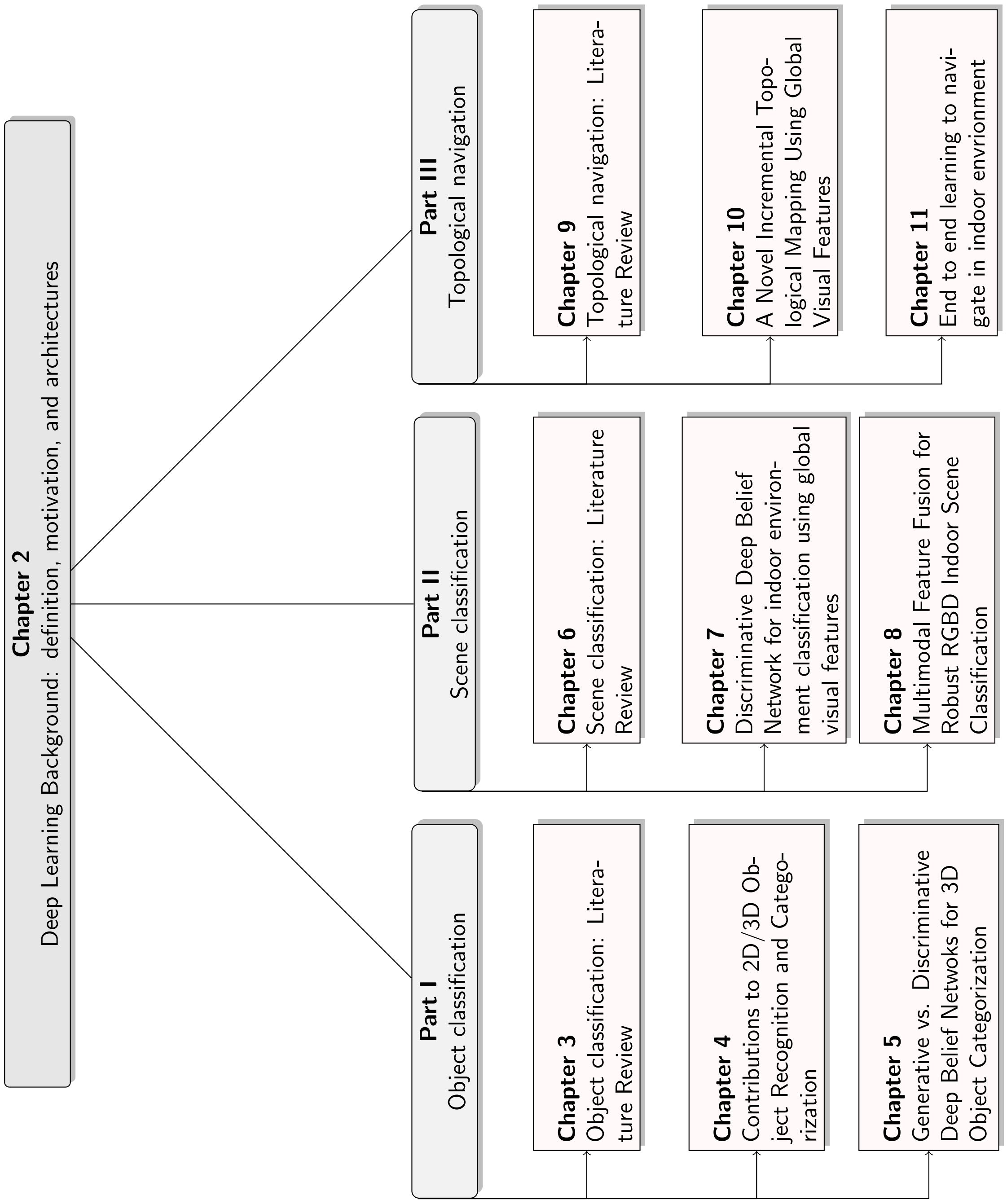

Chapter 2
Deep Learning Background: definition, motivation, and architectures
Part I
Object classification
Chapter 3
Object classification: Literature Review
Chapter 4
Contributions to 2D/3D Object Recognition and Categorization
Chapter 5
Generative vs. Discriminative Deep Belief Netwoks for 3D Object Categorization
Part II
Scene classification
Chapter 6
Scene classification: Literature Review
Chapter 7
Discriminative Deep Belief Network for indoor environment classification using global visual features
Chapter 8
Multimodal Feature Fusion for Robust RGBD Indoor Scene Classification
Part III
Topological navigation
Chapter 9
Topological navigation: Literature Review
Chapter 10
A Novel Incremental Topological Mapping Using Global Visual Features
Chapter 11
End to end learning to navigate in indoor envrionment

Figure 1.4: A roadmap of the thesis.

Chapter 2

Deep Learning Background: definition, motivation, and architectures

I think people need to understand that deep learning is making a lot of things, behind-the-scenes, much better. Deep learning is already working in Google search, and in image search; it allows you to image search a term like "hug".

Geoffrey Hinton

Contents

RESEARCHERS have long dreamed of creating artificial machines that think and act like the human. Over a hundred years ago, when Ada Lovelace [157] built the world's first machine algorithm for an early computing machine that existed only on paper, computer scientists wondered whether such machines might become intelligent. Nowadays, AI covers many practical applications and algorithms that solve the tasks which are easy for humans to perform but hard to describe formally-problems that we solved intuitively; that feel automatic, like recognizing faces in images [117, 135, 195], handwritten digits [103, 121, 123], or spoken words [33, 125, 259]. To handle this challenge, researchers allow computers to learn from experience and understand the world in terms of the hierarchy of concepts where high-level concepts are defined from low-level ones. We can imagine a deep architecture, with many layers, showing how these concepts are built on top of each other. For this reason, the hierarchy of concepts is called in AI *deep learning*. So, what is deep learning? And how to train deep architectures?

In this chapter, we will present a brief history and overview of neural networks that serve as a concise introduction to deep learning models. Moreover, we will introduce some deep learning architectures as well as their training process.

2.1 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) have been developed since 1940 imitating a model of a biological neuron. They provide a practical method for learning real-valued, discrete-valued, and vector-valued functions from examples. Training algorithms such as backpropagation used gradient descent to tune network parameters to best fit a training set of input-output pairs. ANN learning is robust to errors in the training data and has been successfully exploited in applications such as pattern recognition and classification, approximation, optimization, and data clustering. Before we start to describe the technical side of ANNs, it would be useful to briefly introduce the biology system of neural networks.

2.1.1 The components of biological neural network

A nerve cell or neuron is a special biological cell that processes information. It is estimated that there is a huge number of neurons in the human brain, approximately 10^{11}. Two-thirds of this number form about 4-6 mm thick cortex which is assumed to be the center of cognitive processes. Within each neuron, complex biological processes take place, ensuring that it can process signals from other neurons, as well as send its own signals to them. In the description of the neuron components, we will follow the way in which the electrical information takes within the neuron according to Figure 2.1.

- Dendrites: also called dendrite tree, are responsible for receiving the electrical signals from many different sources, which are then transferred into the nucleus of the cell. In other sense, we can say that they are like the ears of the neuron.

- Nucleus (soma): is the cell body of the neuron and is responsible for information processing. After it has received plenty of activating (i.e., stimulating) and inhibiting (i.e., diminishing) signals by synapses or dendrites, the soma accumulates these signals. In the case, if the accumulated signal exceeds a certain value (i.e., threshold value), soma activates an electrical pulse which then is transmitted to the neurons connected to the current one.

- Axon: the pulse is transferred to other neurons by means of the axon. It is just like a cable through which neurons send the information. The axon is electrically isolated in order to achieve better conduction of the electrical signal and it leads to dendrites, which transfer the information to other neurons.
- Synapses: are the connections which transfer the incoming signals from other neurons or cells to a neuron. Such connections can usually be found at the dendrites of a neuron, sometimes also directly at the soma.

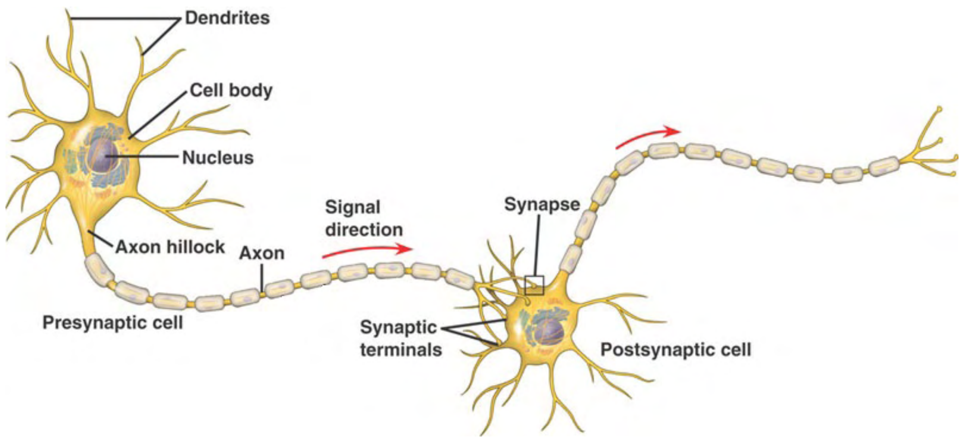


Figure 2.1: Schematic illustration of the basic information processing structure of the biological neuron.

2.1.2 The components of artificial neural network

The artificial neural network is an efficient computing system whose central theme is borrowed from the analogy of biological neurons. ANN consists of simple processing units as well as directed and weighted connections between them. These units also referred to as neurons, are simple processors that operate in parallel. Each neuron is connected with another neuron through a connection link. And every connection link is associated with a weight that being either excitatory or inhibitory.

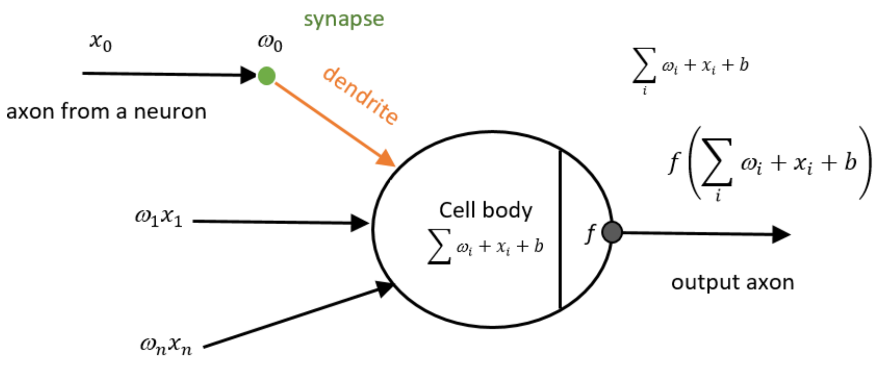


Figure 2.2: Schematic illustration of the basic information processing structure of the artificial neuron.

As shown in Figure 2.2, the coming signals x_0 from the axons interact multiplicatively $\omega_0 x_0$ with the dendrites of the other neuron, based on the synaptic strength at that synapse

ω_0. In the basic model, the dendrites fire the signal to the cell body where they all get summed $\sum_i \omega_i x_i$. Then this sum of the weight signals is added to a baseline or bias value $\sum_i \omega_i x_i + b$. Finally, ANN models the firing rate of the neuron with an activation function f, which represents the frequency of the spikes along the axon.

2.1.3 Biological neural network versus artificial neural network

The study of ANN has been inspired in part by the observation that biological learning systems are built of very complex sites of interconnected neurons. In a rough analogy, ANN is built out of a densely interconnected set of simple units, where each unit takes several inputs and produces a single output. Consequently, there are many complexities to BNN that are not modeled by ANN. For instance, we can put the light on the ANN output which represents a single constant value, while BNN output is a complex time series of spikes. It can be seen that there is a rough analogy between BNN and ANN as well as some differences. Table 2.1 shows the comparison between ANN and BNN based on some mentioned criteria. Whereas, Table 2.2 summarizes the similarities based on the terminology between these two neural networks.

Criteria	**BNN**	**ANN**
Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
Size	10^{11} neurons and 10^{15} interconnections	10^2 to 10^4 nodes (mainly depends on the type of application and network designer)
Learning	They can tolerate ambiguity	Very precise, structured and formatted data is required to tolerate ambiguity
Fault tolerance	Performance degrades with even partial damage	It is capable of robust performance, hence has the potential to be fault tolerant
Storage capacity	Stores the information in the synapse	Stores the information in continuous memory locations

Table 2.1: Differences between biological neural network (BNN) and artificial neural network (ANN).

BNN	**ANN**
Nucleus (soma)	Node
Dendrites	Input
Synapse	Weights or Interconnections
Axon	Output

Table 2.2: Analogy between biological neural network (BNN) and artificial neural network (ANN).

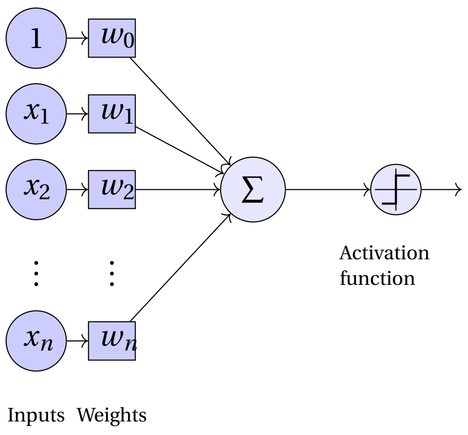


Figure 2.3: Schematic illustration of the SLP model.

2.1.4 Single-Layer Perceptron (SLP)

Developed by Rosenblatt [194] using McCulloch and Pitts model, a single layer perceptron (SLP) is a perceptron having only one trainable weight layer. SLP represents a basic operational unit of ANN which employs supervised learning in order to classify the data into two classes. It consists of a set of inputs along with adjustable weights, but the neuron output is 1 or -1 depending on the threshold. As shown in Figure 2.3, a perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, and sums all the weighted inputs. The most basic activation function used in SLP is a Heaviside step function which has two possible outputs. Then, SLP outputs value 1 if the result is greater than some threshold and value -1 otherwise. More precisely, given the inputs x_1 through x_n, the output $o(x_1,...,x_n)$ computed by the perceptron is calculated as:

$$o(x_1,...,x_n) = \begin{cases} 1 & if \quad \omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n > 0 \\ -1 & otherwise \end{cases} \tag{2.1}$$

Where, ω_i is a real-valued constant or weight, that determines the contribution of input x_i to the perceptron output. The quantity $-\omega_0$ is a threshold that the weighted combination of inputs $\omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n$ must surpass in order for the perceptron to output a value 1. To simplify notation, we imagine an additional constant input $x_0 = 1$, allowing us to write the above inequality as $\sum_{i=0}^{n} \omega_i x_i > 0$. SLP is considered as representing a hyperplane decision surface in the n-dimensional space of points. The perceptron outputs value 1 for points lying on one side of the hyperplane and outputs -1 for points lying on the other side. Those that can be separated are called *linearly separable* sets of points. Minsky and Papert [165] have demonstrated in "Perceptrons" the fact that the SLP can represent all of the primitive boolean functions AND, OR, NAND, and NOR. Unfortunately, some boolean functions cannot be represented by a single perceptron, such as the XOR function which is not linearly separable. To deal with this problem, we can use the kind of Multi-Layer Perceptron (MLP) learned by the *backpropagation algorithm* for expressing a rich variety of non-linear problems.

SLP Learning

The learning problem of SLP is to determine a weight vector which produces the correct 1 or -1 output for each of the given training examples. We consider in this chapter two algorithms that are guaranteed to converge under somewhat different conditions, to somewhat different acceptable hypotheses: the *SLP rule* and the *delta rule* [168].

Learning Rule Rosenblatt's initial perceptron learning rule is a fairly simple algorithm. It represents a typical error correction learning algorithm of single-layer feedforward networks with linear threshold activation function. For learning an acceptable weight vector, the algorithm initializes the weight vector with random weights which will be updated according to the training examples as follows:

- if the perceptron correctly classifies a training example, the algorithm doesn't do anything;
- if the perceptron incorrectly classifies a training example, each of the input weights is updated.

Weights are modified at each step according to the perceptron training rule, which updated the weight w_i associated with input x_i according to Equation 2.2:

$$w_i \leftarrow w_i + \Delta w_i \tag{2.2}$$

with:

$$\Delta w_i = \eta(t - o)x_i \tag{2.3}$$

- t: is the target output for the current training example;
- o: is the output generated by the perceptron;
- η: is a positive constant called the learning rate which moderates the degree to which weights are changed at each step.

Then this process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly. In fact, this algorithm can be proven to converge within a finite number of applications that provided linearly separable training examples. If the data are not linearly separable, convergence is not assured.

Delta Rule In spite the fact that the perceptron rule gets a successful weight vector when the data are linearly separable, it can fail to converge if the data are not. Another training rule, called the delta rule, is provided to overcome this limitation. The delta rule is based on gradient descent to search the hypothesis space of possible weight vectors that best fit the training examples. This training rule considers the task of training an unthreshold perceptron, e.g., a linear unit that corresponds to the first stage of perceptron without a threshold, and for which the output is given by:

$$o(\overrightarrow{x}) = \overrightarrow{w}.\overrightarrow{x} \tag{2.4}$$

Before deriving a weight learning rule for linear units, we specify a training error measure for a weight vector, relative to the training examples. There exist many ways to define this error, one common measure which will turn out to be especially convenient is simply half the squared difference between the target output and the linear unit output.

$$\mathrm{E}(\overrightarrow{w}) = \frac{1}{2}\sum_{d\in \mathrm{D}}(t_d - o_d)^2 \tag{2.5}$$

With:

- D: is the set of training examples;

- t_d: is the target output for training example d;
- o_d: is the output of the linear unit for training example d.

To compute the direction of steepest descent, we calculate the gradient of E with respect to each component of the vector $\vec{w}$.

$$\nabla \mathrm{E}(\vec{w}) = [\frac{\partial \mathrm{E}}{\partial w_0}, \frac{\partial \mathrm{E}}{\partial w_1}, \cdots, \frac{\partial \mathrm{E}}{\partial w_n}] \tag{2.6}$$

As the gradient specifies the direction of steepest increase of E, the training rule for gradient descent is written as follows:

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w} \tag{2.7}$$

Where

$$\Delta \vec{w} = -\eta \Delta \mathrm{E}(\vec{w}) \tag{2.8}$$

We notice that η represents a positive constant called, *learning rate*, which determines the step size in the gradient descent algorithm. Also, the negative sign is added to the equation because the weight vector is moved in the direction that decreases E. This training rule can also be written in its component form:

$$w_i \leftarrow w_i + \Delta w_i \tag{2.9}$$

Where

$$\Delta w_i = -\eta \frac{\partial \mathrm{E}}{\partial w_i} \tag{2.10}$$

To construct a practical algorithm for iteratively updating weights according to Equation 2.10, we need to compute the vector of $\frac{\partial \mathrm{E}}{\partial w_i}$ derivatives that form the gradient by differentiating E from Equation 2.5.

$$\frac{\partial \mathrm{E}}{\partial w_i} = \sum_{d \in \mathrm{D}} (t_d - o_d)(-x_{id}) \tag{2.11}$$

Where x_{id} is the single input component x_i for training example d. Substituting Equation 2.11 into Equation 2.10 yields the weight update rule for gradient descent.

$$\Delta w_i = \eta \sum_{d \in \mathrm{D}} (t_d - o_d) x_{id} \tag{2.12}$$

In summary, the gradient descent algorithm for training linear units is achieved as follows:

- Initialize each w_i to some small random value;
- Apply the linear unit to all training examples;
- Compute Δw_i for each weight according to Equation 2.12;
- Update each weight w_i by adding Δw_i;
- Repeat this process.

We have illustrated two similar algorithms for iteratively learning SLP weights. The key difference between these algorithms is that the SLP training rule updates vector weights based on the error in the thresholded perceptron output, contrary to the delta rule which updates vector weights based on the error in the unthresholded linear combination of inputs.

2.1.5 Multi-Layer Perceptron (MLP)

Proposed by Rumelhart *et al.* [196], the Multi-Layer Perceptron (MLP) consists of a set of neurons that are arranged into more than one layer of variably weighted connections. Like in the SLP, training the MLP follows two phases: the forward phase where the activations are propagated from the given inputs and the current weights to the output, and the backward phase, where the network is updating the weights according to the error; a function of the difference between the outputs and the targets.

MLP learning: going forward

MLP feedforward networks are usually composed by a single layer of hidden neurons inserted between the input and output layers. Such a network is also called the three-layer network. MLP computes multiple real-valued inputs and the first level of weights to calculate the activations of the hidden layer. Then it uses those activations and the next set of weights to calculate the activation of the output layer. MLP just works forward through the network computing linear combination between input weights of two consecutive layers and then putting the output through some non-linear activation function. Mathematically forward can be written as:

$$o(x_1,...,x_n) = \varphi(\sum_{i=0}^{n} \omega_i x_i + b) = \varphi(\mathrm{W}^{\mathrm{T}}\mathrm{X} + b) \tag{2.13}$$

Where

- W: the vector of weights;
- X: the vector of inputs;
- b: the bias;
- φ: the activation function.

Activation functions

The activation functions are non-linear functions that determine the output of neuron. The range of these functions is usually limited between 0 and 1, or between -1 and 1. Early neural networks, including SLP, used a simple threshold function. In case the weighted sum of inputs is less than this threshold, the neuron's output is -1. Otherwise, the output is 1. In some models, such as SLP, the output would be the weighted sum itself when the threshold is exceeded. However, there are great benefits to the function being differentiable, which a threshold certainly is not. The most of current neural networks employ a sigmoid activation function. A sigmoid function is defined as a continuous, real-valued function, whose derivative is always positive, and whose range is bounded. The most commonly used sigmoid function is the *logistic function*. The logistic function ($f(x) = \frac{1}{1+e^{-x}}$) have an output range between 0 and 1, increasing monotonically with its input. It has the useful property that its derivative is easily expressed in terms of its output ($f^{'}(x) = f(x)(1-f(x))$). In most cases, it has been concluded that the exact shape of the function has little effect on the power of the network, though it can have a significant impact on training speed. Kalman and Kwasny[94] make a very eloquent case for choosing the hyperbolic tangent function. They proposed and justified four criteria that an ideal activation function should meet, and showed that only the $tanh$ function meets all four. The authors argue for the use of the hyperbolic tangent, while the exact shape of the sigmoidal makes little difference once the network is trained. It is shown that it possesses particular properties that make it appealing for use during the training process. There exist a number of common activation functions in use with ANN. Figure 2.4 illustrates some popular activation functions.

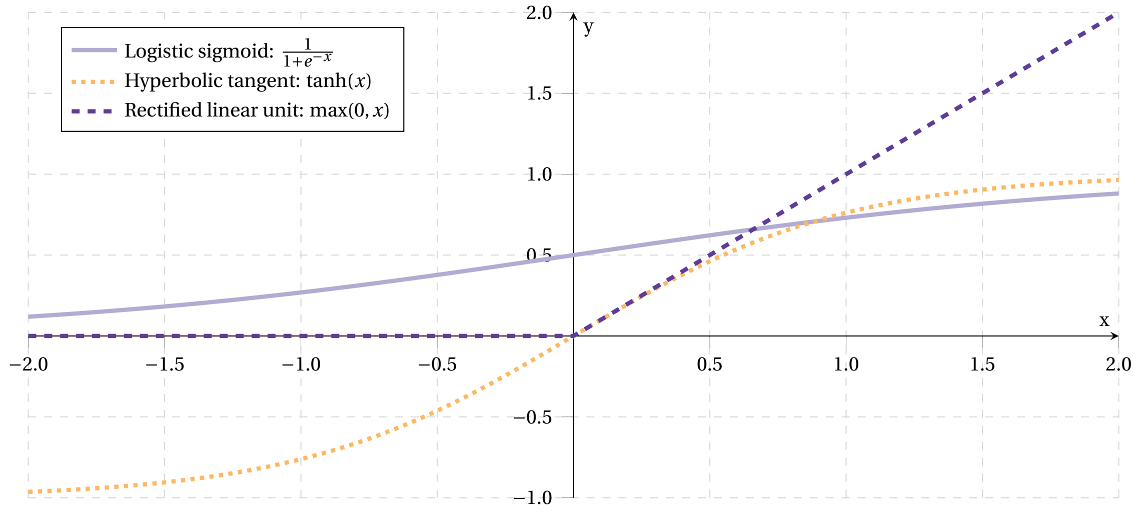


Figure 2.4: Some common activation functions.

MLP learning: going backward (backpropagation of error)

Rumelhart and other collaborators [196, 197] introduced the first practical algorithm for training the MLP feedforward network: Back-Propagation (BP) algorithm. BP uses gradient descent to minimize the squared error between the network output values and the target values for these outputs. The motivation for naming this algorithm as "backpropagation" can be seen when we examine the algorithm for computing the gradient. The output layer errors are successively propagated backward through the network. Since we are considering network with multiple output units, we define the error function E to sum the errors over all of the network output units:

$$\mathrm{E}(t,o) = \frac{1}{2}\sum_{k=1}^{n}(t_k - o_k)^2 \tag{2.14}$$

Where t_k and o_k are respectively target and output values associated with the k^{th} output unit. In Equation 2.14, the value $\frac{1}{2}$ is added to make it easier when we differentiate the function. By differentiating a function, it tells us the gradient of that function, which represents the direction along which it increases or decreases the most. So if we differentiate the error function, we get the gradient of the error. Since the purpose of learning is to minimize the error, following the error function downhill will give us what we want. So, what should be differentiated with respect to?

In general, there are only three things: the inputs, the activation function which decides whether or not the node fires, and the weights. The first and the second are out of our control once this algorithm starts running, so only the weights matter, and therefore, they are what we differentiate with respect to. Now, it remains only to be seen what should be differentiated. In section 2.1.5, we noticed that sigmoid functions are differentiable so that we can compute the gradient. They also have another nice property, which that their derivative has a nice form. Therefore, activation function is the thing which we can differentiate so that when we change the weights, we do it following the direction that is downhill for the error, which means that we know we are improving the error function of the network. At the output, the errors are computed as the sum-squared difference between the targets and the outputs. After that, we compute the gradient of these errors and use them to decide how much to update each weight in the network. Firstly, we will do that for the nodes connected to the output layer, and after those have been updated, we work backward through the network until we get back to the inputs again. We can encounter two problems:

1. for the output neurons, we don't know the inputs;
2. for the hidden neurons, we don't know the targets.

So the error at the output can be computed, but since the inputs were that caused it are unknown, the algorithm cannot update those weights. One solution to this problem is to use the chain rule differentiation. The chain rule shows that if the algorithm wants to discover how the error changes as it varies the weights, it can know how the error changes as it varies the inputs to the weights, and multiply this by how those input values change as it varies the weights. This is important because it can write the activations of the output nodes in terms of the activations of the hidden nodes and the output weights, and then it can send the error calculations back through the network to the hidden layer in order to decide what the target outputs were for those neurons.

In summary, the backward algorithm computes the gradients of the errors with respect to the weights, so that it changes the weights to go downhill, which makes the errors smaller. To do that, the algorithm differentiates the error function with respect to the weights, and since it cannot do that directly, it has to apply the chain rule and differentiate with respect to things that it knows. This leads to two different update functions, one for each of the weight sets, and the algorithm just applies these backwards through the network, starting at the outputs and ending up back at the inputs.

MLP learning algorithm

MLP has been applied to solve some diverse and difficult problems by training it in a supervised manner with a popular algorithm called backpropagation. Algorithm 1 provides a description of the basic MLP algorithm. An input vector which represents the feature vector of a specific problem or application is put into the input nodes of the MLP network. The MLP algorithm suggests that the weights are initialized to small random values (i.e., positive and negative). Each neuron is getting up from n different places, in other words, either input nodes when the neuron is in the hidden layer, or hidden neurons when it is in the output layer. When the values of these inputs are considered as having uniform variance, the typical input to the neuron will be $\omega\sqrt{n}$, with ω is the initialization value of the weights. However, a common trick is to set the weights in the range $\frac{-1}{\sqrt{n}} < \omega < \frac{1}{\sqrt{n}}$, where n is the number of nodes in the input layer to those weights. Random values are used in this range so that the learning starts off from different places of each run, and also keep them all about the same size. This is known as *uniform learning* which is important because all of the weights will reach their final values at about the same time. Basically, backpropagation training consists of two passes through the different layers of the network: a forward pass and a backward pass (for more details we can refer to [149]). In the forward pass, the inputs are fed forward through the network as follows:

- compute the activations of the hidden neurons; the inputs and the first-layer weights (i.e., labeled in the algorithm as v) are used to decide whether the hidden nodes fire or not;

- compute the activations of the output neurons; the activations of the hidden neurons and the second-layer weights (i.e., labeled in the algorithm as w) are used to decide if the output neuron fire or not.

We notice that i is the index over the input nodes, j is the index over the hidden layer neurons, and k is the index over the output neurons. The activation function $g(.)$ represents the sigmoid function. During the backward pass, the error is computed as the sum of squares difference between the network outputs and the targets. After that, this error is fed backward through the network in order to first update the second-layer weights by using the δ_o errors, and then afterward, the first layer weights by using the δ_h errors until learning stops.

Algorithm 1 Multi-Layer Perceptron Algorithm.

```
1: Initialization
2:    Initialize all weights to small (positive and negative) random values
3: Training
4:    repeat:
5:     for each input vector:
6:       Forward phase
7:         compute the activation of each neuron $j$ in the hidden layer using:
8:            $h_j = \sum_i x_i v_{ij}$
9:            $a_j = g(h_j) = \frac{1}{1+exp(-\beta h_j)}$
10:        compute the activation of each neuron $k$ in the output layer using:
11:           $h_k = \sum_j a_j w_{jk}$
12:           $o_k = g(h_k) = \frac{1}{1+exp(-\beta h_k)}$
13:      Backward phase
14:        compute the error in the output using:
15:           $\delta_{ok} = (t_k - o_k) o_k (1 - o_k)$
16:        compute the error in the hidden layer using:
17:           $\delta_{hj} = a_j (1 - a_j) \sum_k w_{jk} \delta_{ok}$
18:        update the output layer weights using:
19:           $w_{jk} \leftarrow w_{jk} + \eta \delta_{ok} a_j^{hidden}$
20:        update the hidden layer weights using:
21:           $v_{ij} \leftarrow v_{ij} + \eta \delta_{h_j} x_i$
22:      Until learning stops
23: Recall
24:      Use the Forward phase
```

MLP with many hidden layers

MLP with many hidden layers is often referred as a deep neural network (DNN) which represents a basic example of the deep learning architectures. In the 1980s, BP algorithm has been popularized in the community of researchers to learn the parameters of this type of networks. However, BP alone did not work well in practice for learning networks with more than a small number of hidden layers. The presence of local optima and other optimization challenges in the non-convex objective function of the deep networks are the major source of difficulties in the learning process. BP is based on local gradient information and starts usually at some random initial points. It often gets trapped in poor local optima when the batch-mode or even stochastic gradient descent BP algorithm is used. This challenge becoming increasingly complicated as the depth of the networks increases. However, until 2006, researchers didn't know how to train ANN to surpass more traditional approaches, except for a few specialized applications such as speech recognition and natural language processing. Hinton *et al.* [81, 82] suggested the procedures for learning in MLP, and these techniques are now recognized as "deep learning", or more particularly a class of deep generative models, called Deep Belief Network (DBN), which is composed of a stack of Restricted Boltzmann Machines (RBMs).

2.2 Deep Learning

Deng *et al.* [46] defined deep learning as a class of machine learning systems that process information for unsupervised or supervised feature extraction using numerous layers of nonlinear processing. Those features are later employed for pattern analysis and classification. Deep architectures attempt to extract features in various stages of abstraction for enabling a system to pick up complex functions that map the input data to the output directly. The stages in these statistical models consist of discrete levels of concepts where lower-level concepts define the higher-level ones [20]. Serre *et al.* [214] showed the evidence that the brain of a mammal is organized as a deep architecture. A specified input is characterized by various levels of abstraction where every level relates to a diverse area of cortex. Researchers used the deep architecture concept in neural networks for training new deep MLP networks which are stimulated by the biological depth of the brain. Such deep models involve numerous layers and parameters that require being learned through the complex process. To deal with this problem, Hinton *et al.* [81] suggested a DBN with multiple layers of hidden units. DBN is a graphical model comprising undirected networks at the top hidden layers and directed networks in the lower ones. The learning algorithm uses greedy layer-wise training by stacking RBMs. It comprises a hidden layer for modeling the probability distribution of visible variables. This model is used for several classification tasks such as object classification, speech recognition, and phone recognition [169]. In general, deep learning architectures can be broadly classified into three main categories [45]:

1. generative deep architectures: the aim is to characterize the high-order correlation properties of the visible data for pattern analysis or synthesis purposes, and/or characterize the joint statistical distributions of the visible data and their associated classes;

2. discriminative deep architectures: the aim is to directly provide discriminative power for pattern classification, often by characterizing the posterior distributions of classes conditioned on the visible data;

3. hybrid deep architectures: the aim is to combine the power of discrimination with the outputs of generative architectures via better optimization or/and regularization.

2.2.1 Energy-Based Models (EBMs)

Energy-Based Models (EBMs) associate scalar energy (i.e., a measure of compatibility) to each configuration of the variables of interest. The inference (i.e., prediction) consists in setting the value of observed variables and finding values of the remaining variables that minimize the energy. The learning corresponds to modifying that energy so that its shape has desirable properties, in which observed configuration of the variables are given lower energies than unobserved ones [20]. Energy-based probabilistic models may define a probability distribution through an energy function, as follows:

$$p(x) = \frac{e^{-\mathrm{E}(x)}}{\mathrm{Z}} \tag{2.15}$$

The normalizing factor $\mathrm{Z} = \sum_x e^{-\mathrm{E}(x)}$ presents the partition function by analogy with physical systems. In many cases of interest, x possesses many component variables x_i that are not observed simultaneously, or that are introduced like non-observed variables to increase the expressive power of the model. So, considering an observed part x and a hidden part h, the probability distribution becomes as follows:

$$p(x) = \sum_h p(x,h) = \sum_h \frac{e^{-\mathrm{E}(x,h)}}{\mathrm{Z}} \tag{2.16}$$

2.2.2 Restricted Boltzmann Machines (RBMs)

Boltzmann Machines (BMs) are a particular type of energy-based model with hidden variables [2]. They have been introduced as bidirectionally connected networks of stochastic processing units, which can be interpreted as a neural network model. BMs can also be regarded as undirected graphical models, known as Markov Random Fields (MRFs). In general, BMs are used to learn important aspects of an unknown probability distribution despite their difficulty and their computing time. Nevertheless, the learning problem can be simplified by imposing restrictions on the network topology (i.e., visible-visible and hidden-hidden), which provide Restricted Boltzmann Machines (RBMs).

As undirected graphical models, RBMs represent a probability distribution that can be used in an unsupervised learning problem to model some distributions over some inputs. Given a set of training data, learning corresponds to adjusting the model parameters of the RBMs such that the represented probability distribution fits the training data as well as possible. The RBMs then form a model of the distribution underlying the training data [57]. RBMs can be trained in various ways and used as a generative model, a discriminative model, a feature extractor, or building blocks of deep architectures.

- RBMs as generative models: RBMs are used for drawing samples from the learned distribution.
- RBMs as a discriminative model: RBMs is trained on the joint distribution of inputs and labels, one can sample the missing label for a represented data from the distribution or assign a new data to the class with the highest probability under the model.
- RBMs as feature extractors: RBMs consist of two types of units. A layer of visible units which correspond to the components of the inputs, and a layer of hidden units which capture dependencies between the visible neurons. After training, the expected states of the hidden variables given input can be interpreted as the features extracted from this input pattern.
- RBMs as building blocks of deep architectures: RBMs are proposed as building blocks of multi-layer learning deep architectures called DBNs. The idea behind is that the hidden neurons extract pertinent features from the visible neurons. These features can work as the input to another RBM. By stacking RBMs in this way, the model can learn features for a high-level representation.

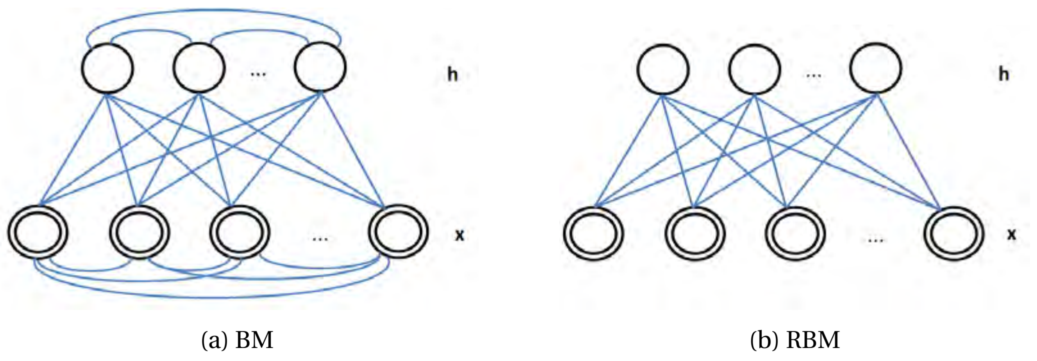


Figure 2.5: BM and RBM models. (b) the joints between hidden units and also between visible units are disconnected.

Generative Restricted Boltzmann Machines (GRBMs)

Restricted Boltzmann Machines (RBMs) are a specific category of energy based model which include hidden variables. RBMs are restricted in the sense so that no hidden-hidden or variable-variable connections exist. The architecture of a generative RBM is illustrated in Figure 2.5b. RBMs are a parameterized generative stochastic neural network which contain stochastic binary units on two layers: the visible layer and the hidden layer.

1. Visible layer (the first layer): it contains visible units x that correspond to the components of an observation;
2. Hidden layer (the second layer): it contains hidden units h that model dependencies between the components of observations.

The stochastic nature of RBMs results from the fact that the visible and hidden units are stochastic. The units are binary, i.e. $x_j \in \{0,1\}^m$, $h_i \in \{0,1\}^n$.

$$\mathrm{E}(x,h) = -\sum_{j=1}^{m} b_j x_j - \sum_{i=1}^{n} c_i h_i - \sum_{j=1}^{m}\sum_{i=1}^{n} \mathrm{W}_{ij} x_j h_i \tag{2.17}$$

where:

- W represents the symmetric interaction term between visible units x and hidden units h;
- b and c are vectors that store the visible (input) and hidden biases (respectively).

Equation 2.17 can be used to define a Gibbs probability distribution of the form

$$p(x) = \sum_{h} \frac{e^{-\mathrm{E}(x,h)}}{\mathrm{Z}} \tag{2.18}$$

The normalization constant is $\mathrm{Z} = \sum_{x,h} e^{-\mathrm{E}(x,h)}$ and the energy function of an RBM is defined as:

$$\mathrm{E}(x,h) = -b^{'}x - c^{'}h - h^{'}\mathrm{W}x \tag{2.19}$$

The RBM topology structure has connections only between the layer of hidden and the layer of visible units, but not between two units of the same layer. In terms of probability, visible and hidden units are conditionally independent given one-another. Using this property, we can write:

$$p(h \mid x) = \prod_{i} p(h_i \mid x) \tag{2.20}$$

$$p(x \mid h) = \prod_j p(x_j \mid h) \tag{2.21}$$

RBM can be regarded as a stochastic neural network, where the nodes and edges correspond to neurons and synaptic connections, respectively. As feedforward neural network, RBM consists of one layer of non-linear processing units. From this perspective, RBM can be interpreted as a deterministic function that maps an input $x \in \{0,1\}^m$ to $y \in \mathbb{R}^n$ with $y_i = \mathrm{P}(h_i = 1 \mid x)$.

We obtain from Equations 2.18 and 2.19, a probabilistic version of the usual neuron activation function:

$$p(h_i = 1 \mid x) = \frac{e^{c_i + \mathrm{W}_i x}}{1 + e^{c_i + \mathrm{W}_i x}} = sigm(c_i + \mathrm{W}_i x) \tag{2.22}$$

$$p(x_j = 1 \mid h) = \frac{e^{b_j + \mathrm{W}'_j h}}{1 + e^{b_j + \mathrm{W}'_j h}} = sigm(b_j + \mathrm{W}'_j h) \tag{2.23}$$

Restricted Boltzmann Machines have been used widely and effectively in modeling distributions over binary-valued input data. Recently, due to the growth of machine learning applications such as image classification and speech recognition [99], many works [82, 204] have extended the standard RBM to Gaussian-Bernoulli RBM which is very suitable to real-valued input data (e.g., image pixels or word-count vectors). In addition to reviewing the basic version of RBM, we will also review Gaussian-Bernoulli RBM which models real-valued inputs that are very appropriate to our input data.

Gaussian-Bernoulli Restricted Boltzmann Machines (GBRBMs)

To model distributions over real-valued input data, such as image pixels [273] or word-count vectors, we can use Gaussian-Bernoulli Restricted Boltzmann Machine variant [82]. In particular, consider modeling visible real-valued units $x_j \in \mathbb{R}^m$, and let $h_i \in \{0,1\}^n$ be stochastic binary hidden variables. The energy of the joint state $\{x, h\}$ of the Gaussian RBM is defined as follows:

$$\mathrm{E}(x,h) = -\sum_{j=1}^{m} \frac{(b_j - x_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n} c_i h_i - \sum_{j=1}^{m}\sum_{i=1}^{n} \mathrm{W}_{ij} h_i \frac{x_j}{\sigma_j} \tag{2.24}$$

Where b_j and c_i are biases corresponding respectively to visible and hidden units, W_{ij} are the connecting weights between the visible and hidden units, and σ_j is the standard deviation associated with Gaussian visible units x_j. Since GBRBM has a bipartite structure, the visible units given the hidden units are conditionally independent, and the probability of each visible unit is given by:

$$p(x_j = x \mid h) = \mathrm{N}(x \mid b_j + \sum_i h_i \mathrm{W}_{ij}, \sigma_j^2) \tag{2.25}$$

Where $\mathrm{N}(. \mid \mu, \sigma^2)$ denotes the Gaussian probability density function with mean μ and standard deviation σ. Similarly, the hidden units are conditionally independent, and their probabilities are given by:

$$p(h_i = 1 \mid x) = sigm(c_i + \sum_j \mathrm{W}_{ij} \frac{x_j}{\sigma_j^2}) \tag{2.26}$$

Discriminative Restricted Boltzmann Machines (DRBMs)

RBMs have been used as generative models of many different types of data. They use a layer of hidden variables to model a distribution over visible variables. Such models are usually trained to only model the inputs of the classification task. Moreover, they can model the joint distribution of the inputs and associated target classes like in the last layer of a DDBN (in Section 2.2.3). In our contributions, we are interested in such joint models for a classification application. In general, Hinton [80] introduces three manners of using discriminative RBMs:

1. using the hidden variables learned by the RBM as the inputs for some discriminative method;

2. training a separate RBM on each class;

3. training a joint density model using a single RBM that has two sets of visible units.

The last method is proposed by Larochelle and Bengio[116] and aims to train a density model using a single RBM that has two sets of visible units. Figure 2.6 shows the joint distribution for the inputs x and associated target classes y [17].

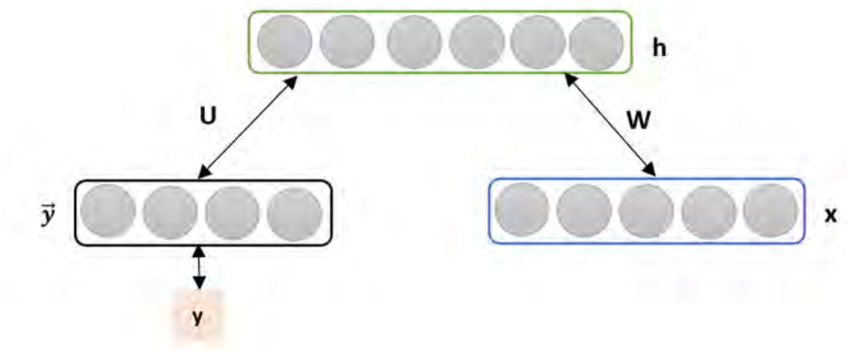


Figure 2.6: RBM modeling the joint distribution of inputs x and target class y (represented as one-hot vector by $\vec{y}$) from [17] . Hidden units are denoted by h.

RBM is a parametric model of the joint distribution between a layer of hidden variables $h = (h_1, ..., h_n)$ and the visible variables of the inputs $x = (x_1, ..., x_d)$ and the target y, that is defined as:

$$p(y, x, h) \propto e^{-\mathrm{E}(y,x,h)} \tag{2.27}$$

where

$$\mathrm{E}(y, x, h) = -h^{'}\mathrm{W}x - b^{'}x - c^{'}h - d^{'}\vec{y} - h^{'}\mathrm{U}\vec{y} \tag{2.28}$$

with $\Theta = (\mathrm{W}, b, c, d, \mathrm{U})$ is the set of parameters and $\vec{y} = (1_{y=j})_{j=1}^{c}$ for C classes. We consider that the input variables x are binary. The conditional distributions between layers are defined as:

$$p(x \mid h) = \prod_{j} p(x_j \mid h) \tag{2.29}$$

$$p(x_j = 1 \mid h) = sigm(b_j + \sum_{i} \mathrm{W}_{ji} h_i) \tag{2.30}$$

$$p(y \mid h) = \frac{e^{d_y} + \sum_i \mathrm{U}_{iy} hi}{\sum_{y^*} e^{d_{y^*}} + \sum_i \mathrm{U}_{iy^*} hi} \tag{2.31}$$

Equations (2.30) and (2.31) indicate that the hidden units are supposed to capture predictive information about the input vector x and the target class y. Similarly, the conditional distribution of hidden units given inputs and target class, $p(h \mid y, x)$, is defined as follows:

$$p(h \mid y, x) = \prod_i p(h_i \mid y, x) \tag{2.32}$$

$$p(h_i = 1 \mid y, x) = sigm(c_i + \mathrm{U}_{iy} + \sum_j \mathrm{W}_{ji} x_j) \tag{2.33}$$

RBM learning

RBMS have been used as generative models of many different types of data and applications [83, 170]. Recently, their most important use is building blocks for the multi-layer learning systems called DBNs. The success of RBMs raises the issue of how best to train them. All training algorithms for RBMs approximate the log-likelihood gradient given some data then perform gradient ascent/descent on these approximations. The derivative of the log-likelihood is obtained from Equation 2.18:

$$\frac{\partial \log p(x)}{\partial \theta} = -\sum_h p(h \mid x) \frac{\partial \mathrm{E}(x,h)}{\partial \theta} + \sum_{x,h} p(x,h) \frac{\partial \mathrm{E}(x,h)}{\partial \theta} \tag{2.34}$$

Since θ is an element in the set of parameters (W, c, b), Equation 2.34 can be split into three part:

$$\begin{aligned} \frac{\partial \log p(x)}{\partial \mathrm{W}_{ij}} &= \langle x_j h_i \rangle_{data} - \langle x_j h_i \rangle_{model} \\ \frac{\partial \log p(x)}{\partial b_j} &= \langle x_j \rangle_{data} - \langle x_j \rangle_{model} \\ \frac{\partial \log p(x)}{\partial c_i} &= \langle h_i \rangle_{data} - \langle h_i \rangle_{model} \end{aligned} \tag{2.35}$$

With $\langle x_j h_i \rangle_{data}$ and $\langle x_j h_i \rangle_{model}$ represent the expected values with respect to data and model distribution accordingly. The first term of Equation 2.34 can be computed analytically in GRBM since $p(h \mid x)$ of the hidden states can be factorized. In contrast, the second term represents the gradient of the partition function and reffers to the model expectation. It is too burdensome for direct computation and requires to be approximated. For this reason, Markov Chain Monte-Carlo (MCMC)[173] sampling methods are employed in order to approximate this expectation by samples from the model distribution. These samples can be generated by Gibbs sampling which requires running the Markov chain "long enough" to ensure convergence to stationarity.

RBM defines a distribution over all of its variables, they exist several strategies that can be used to train it. The most common one is called the generative training. Consider a training set $\tau = \{(x_j, y_j)\}$ of N_τ pairs of the j^{th} example an input vector x_j and a target class $y_j \in 1, \ldots, \mathrm{C}$. To train a generative model on such data we consider minimization of the negative log-likelihood:

$$\mathscr{L}_{generative}(\tau) = -\sum_{j=1}^{\mathrm{N}_\tau} log(y_j, x_j) \tag{2.36}$$

In order to minimize the negative log-likelihood (Equation 2.36), we apply the gradient with respect to the model parameters. The exact gradient, for any $\theta \in \Theta$ parameter can be defined as follows:

$$\frac{\partial log p(y_j, x_j)}{\partial \theta} = -\mathbb{E}_{h|y_j x_j}\left[\frac{\partial}{\partial \theta}\mathrm{E}(y_j, x_j, h)\right] + \mathbb{E}_{y,x,h}\left[\frac{\partial}{\partial \theta}\mathrm{E}(y, x, h)\right] \tag{2.37}$$

The first expectation is computed exactly, while the second one (i.e., the expectation under the model distribution) is intractable. $p(y,x)$ is intractable, but it is possible to compute $p(y \mid x)$, sample from it, or choose the most probable class. For reasonable numbers of classes C (i.e., over which we must sum) as defined by Salakhutdinov *et al.* [205], this conditional distribution can be computed exactly and efficiently as follows:

$$p(y|x) = \frac{e^{d_y} \Pi_{i=1}^{m}(1 + e^{c_i + \mathrm{U}_{iy} + \Sigma_j \mathrm{W}_{ji} x_j})}{\Sigma_{y^*} e^{d_{y^*}} \Pi_{i=1}^{m}(1 + e^{c_i + \mathrm{U}_{iy^*} + \Sigma_j \mathrm{W}_{ji} x_j})} \tag{2.38}$$

To train the discriminative model, it can then be advantageous to optimize directly $p(y \mid x)$ instead of $p(y,x)$:

$$\mathscr{L}_{discriminative}(\tau) = -\sum_{j=1}^{\mathrm{N}_\tau} log(y_j \mid x_j) \tag{2.39}$$

A DRBM can be trained, since $p(y \mid x)$ can be computed exactly, we can compute the exact gradient as follows:

$$\frac{\partial \log p(y^{(j)}|x^{(j)})}{\partial \theta} = \sum_i sigm(o_{y^{(j)},i}(x^{(j)}))\frac{\partial o_{y^{(j)},i}(x^{(j)})}{\partial \theta} - \sum_{i,y^*} sigm(o_{y^*i}(x^{(j)}))p(y^*|x^{(j)})\frac{\partial o_{y^*i}(x^{(j)})}{\partial \theta} \tag{2.40}$$

where, $o_{y,i}(x) = c_i + \sum_k \mathrm{W}_{i,k} x_k + \mathrm{U}_{i,y}$. For GBRBM, the derivative of the log-likelihood with respect to W, b, and c parameters is obtained from Equation 2.24:

$$\begin{aligned}\frac{\partial \log p(x)}{\partial \mathrm{W}_{ij}} &= \langle\frac{1}{\sigma_j} x_j h_i\rangle_{data} - \langle\frac{1}{\sigma_j} x_j h_i\rangle_{model}\\ \frac{\partial \log p(x)}{\partial b_j} &= \langle\frac{1}{\sigma_j^2} x_j\rangle_{data} - \langle\frac{1}{\sigma_j^2} x_j\rangle_{model}\\ \frac{\partial \log p(x)}{\partial c_i} &= \langle h_i\rangle_{data} - \langle h_i\rangle_{model}\end{aligned} \tag{2.41}$$

As discussed in GRBM paragraph, the second term of Equation 2.41 is intractable and should be approximated. In practice, instead of learning σ^2, the standard deviation is assumed to be constant throughout the training.

In all RBM models including, GRBM, BGRBM, and DRBM, sampling methods are used for gradient estimation which requires samples from the model that has been trained. While in an RBM each unit in a layer is independent of other units in other layers, Gibbs sampling is a proper method but it requires a large computing time. For this reason, some methods are proposed such as Contrastive Divergence (CD), Persistent Contrastive Divergence (PCD), and Free Energy in Persistent Contrastive Divergence (FEPCD).

- Contrastive Divergence (CD)

To get a tractable approximation of the second expectation in Equations 2.34, 2.40, and 2.41, we use some algorithms to approximately sample from the model. The Contrastive Divergence (CD) algorithm is a standard way to do this [79]. The key idea of k-step CD learning algorithm is quite simple. Instead of approximating the second term in the log-likelihood gradient by a sample from the GRBM, DRBM, or BGRBM distribution, the algorithm runs a

Gibbs chain for only k-steps (usually k=1). The Gibbs chain is initialized with a training example $x^{(0)}$ of the training set data and yields the sample $x^{(k)}$ after k steps. In every step t, CD consists of sampling $h^{(t)}$ from $p(h \mid x^{(t)})$ and subsequently sampling $x^{(t+1)}$ from $p(x \mid h^{(t)})$. The gradient of the log-likelihood for one training pattern $x^{(0)}$ is then approximated by:

$$\mathrm{CD}_k(\theta, x^{(0)}) = -\sum_h p(h \mid x^{(0)}) \frac{\partial \mathrm{E}(x^{(0)}, h)}{\partial\theta} + \sum_h p(h \mid x^{(k)}) \frac{\partial \mathrm{E}(x^{(k)}, h)}{\partial\theta} \tag{2.42}$$

A batch version of CD-k can be illustrated in Algorithm 2. In batch learning, the complete training data set S is used to compute or approximate the gradient in every step.

Algorithm 2 K-step Contrastive Divergence

```
1:  **Input**: RBM ($X_1, ..., X_m, H_1 ..., H_n$), training batch S
2:  **Output**: gradient approximation $\Delta w_{ij}, \Delta b_j$ and $\Delta c_i$ **for** $i = 1, ... n$ and $j = 1, ..., m$
3:  init $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, ... n, j = 1, ..., m$
4:  **for** all the $x \in$ S do
5:    $x^{(0)} \leftarrow x$
6:    **for** $t = 0, ..., k-1$ do
7:      **for** $i = 1, ..., n$ do sample $h_i^{(t)} \sim p(h_i | x^{(t)})$
8:      **for** $j = 1, ..., m$ do sample $x_j^{(t+1)} \sim p(x_j | h^{(t)})$
9:    **for** $i = 1, ..., n$ and $j = 1, ..., m$ do
10:     $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 | x^{(0)}).x_j^{(0)} - p(H_i = 1 | x^{(k)}).x_j^{(k)}$
11:   **for** $j = 1, ..., m$ do
12:     $\Delta b_j \leftarrow \Delta b_j + x_j^{(0)} + x_j^{(k)}$
13:   **for** $i = 1, ..., n$ do
14:     $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 | x^{(0)}) - p(H_i = 1 | x^{(k)})$
```

CD-k learning is the basis for a very effective approach for learning random fields. It has been successfully applied to training RBMs [79, 81] in various applications. However, the convergence to a Maximum Likelihood (ML) estimates is not always guaranteed. Mackay [143] provided some examples of the convergence, but he used unusual sampling operators such as drift-and-diffuse, swirl, flip, and star-trek operators. Yuille [275] gives a condition under which the algorithm is able to converge to the optimal solution. Carreira *et al.* [36] showed that CD learning provides biased estimates in general, but that the bias is typically very small. More recently, Merino *et al.*[158] argued that CD has a number of shortcomings, and its approximation to the gradient has several drawbacks. Since it is not able to assign large enough probabilities to the examples in the training set.

- *Persistent Contrastive Divergence (PCD)*

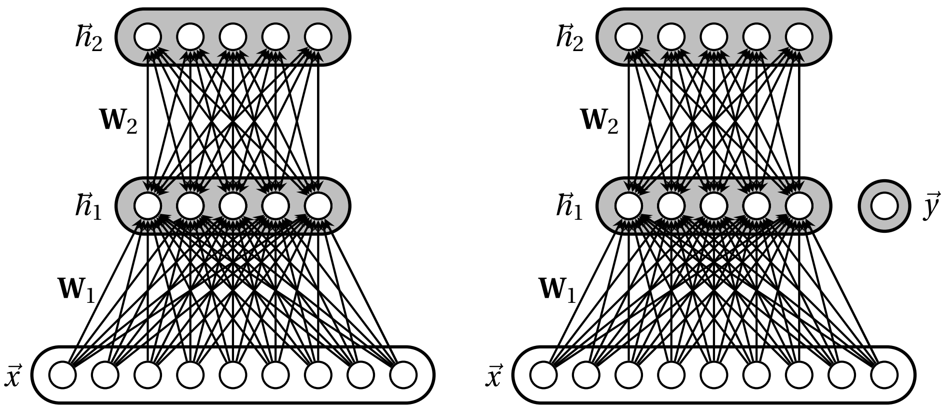
Since CD sampling has some disadvantages and is not precise, the PCD method is proposed to use just last chain state in the last update step. All model parameters are changed in each step, but can receive good samples from a model distribution with a few Gibbs sampling steps because the model parameters change slightly [242].

- *Free Energy in Persistent Contrastive Divergence (FEPCD)*

In PCD sampling, many persistent chains can be run in parallel, and we will refer to the current state in each of these chains as a "fantasy" particle. However, chain selection is blind and the best one may not be selected. Recently, Keyvanrad and Homayounpour [100] proposed a new sampling method that defines a criterion for goodness of a chain. This method uses free energy as a criterion to obtain elite samples from a generative model that can more accurately compute the gradient of the log probability of training data.

2.2.3 Deep Belief Networks (DBNs)

Deep Belief Network (DBN) is the probabilistic generative model with many layers of stochastic and hidden variables. Hinton *et al.* [81] introduced the motivation for using a deep network versus a single hidden layer (i.e., a DBN vs. an RBM). The power of deep networks is achieved by having more hidden layers. However, one of the major problems for training deep network is how to initialize the weights W between the units of two consecutive layers ($l-1$ and l), and the bias b of layer l. Random initializations of these parameters can cause poor local minima of the error function resulting in low generalization. For this reason, Hinton *et al.* introduced a DBN architecture based on a training sequence of RBMs. Afterward, we will present two general types of DBN architectures that are used in our work: Generative Deep Belief Network (GDBN) and Discriminative Deep Belief Network (DDBN). Both types use the greedy layer-wise algorithm. The terms generative and discriminative refer to the nature of RBMs used in each architecture. The first one is composed of GRBMs and adds a final layer of variables that represent the desired outputs then performs a purely discriminative fine-tuning phase using backpropagation. We refer to this architecture as "GDBN" or also "BP-DBN" (Back-Propagation DBN). While the second one consists of GRBMS and DRBM in the last layer. We refer to this architecture as "DDBN".



(a) GDBN (b) DDBN

Figure 2.7: DBN architectures with one visible layer x and two hidden layers h_1 and h_2.

Generative Deep Belief Network (GDBN)

GDBN training algorithm consists of two stages: (i) layer-wise generative pre-training, and (ii) fine-tuning the model. In the generative pre-training stage, DBN trains sequentially as many RBMs as the number of hidden layers that constitute its architecture, i.e., for a DBN architecture with l hidden layers, the model has to train l RBMs. For the first RBM, the inputs consist of the DBN's input layer (i.e., visible units) and the first hidden layer. For the second RBM, the inputs consist of the hidden unit activations of the previous RBM and the second hidden layer. The same holds for the remaining RBMs to browse through the l layers. The layer wise training starts from the first RBM. The units on both first hidden layer h^1 and visible layer x are trained to model a given training inputs $\mathrm{D}_0 = \{x^{(m)} \mid m = 1, 2, \cdots, \mathrm{M}\}$. The training refers to estimating the RBM parameters, weights, and biases using CD, PCD, and FEPCD algorithms. After estimating the parameters θ^1 of the first RBM, a set of samples D_1 of the first hidden layer states h_1 is drawn from $\mathrm{Q}(h^1 \mid x^m, \theta^1)$ with $m = 1, 2, \cdots, \mathrm{M}$, for training of the next level RBM. We denote posteriors with $\mathrm{Q}(.)$ because they only approximate the true posterior that is also dependent on the above hidden layers. The posterior $Q(h_1 \mid \theta^1)$ from which the samples are effectively collected is given in Equation 2.43:

$$Q(h_1 \mid \theta^1) = \frac{1}{M} \sum_{m=1}^{M} Q(h_1 \mid x^{(m)}, \theta^1) \tag{2.43}$$

After that, the algorithm trains the second RBM by using the set D_1 that contains samples of the hidden layer h_1 states (i.e., as an input to the second RBM). The second RBM which is composed of the hidden layers h_1 and h_2, is trained to model the posterior $Q(h_1 \mid \theta^1)$ through the samples. After training the second RBM, and thus estimating the corresponding RBM parameters θ^2, samples from posterior $Q(h_2 \mid \theta^2)$ are collected for training the next RBM level. The generative pre-training proceeds subsequently by treating each pair of consecutive hidden layers, h^{l-1} and h^l as an RBM and training them to model the lower level posterior $Q(h_{l-1} \mid \theta^{l-1})$.

Algorithm 3 Greedy layer-wise learning procedure for DBN

```
1: Fix the parameters $\theta^1$ of the first-layer RBM to data;
2: Fix the parameter vector $\theta^1$, and use samples $h^1$ from $Q(h^1) \mid \theta^1)$ as the data for training
   the next layer of binary features with an RBM;
3: Fix the parameters $\theta^2$ that define the second layer of features, and use the samples $h^2$
   from $Q(h^2) \mid \theta^2)$ as the data for training the third layer of binary features;
4: Proceed recursively for the next layers.
```

After the model performs Algorithm 3, a good initialization of the biases and the hidden weights of the DBN is obtained. At this stage, the model should determine the weights from the last hidden layer for the outputs. To obtain a successfully supervised learning, the model "fine-tunes" the resulting weights of all layers together. Figure 2.7a illustrates a generative DBN architecture with one visible layer and two hidden layers.

Discriminative Deep Belief Network (DDBN)

DDBN architectures have been proposed for different applications [138, 286]. Here, we introduce a learning algorithm; Discriminative Deep Belief Network (DDBN) based on Discriminative Restricted Boltzmann Machine (DRBM) as defined in [100].

DBN aims at letting every RBM model in the structure to obtain a diverse representation of data. In other words, after RBM is trained, the activity values from the hidden units act as the training data for a higher-level RBM learning. In DDBN, we need to use a DRBM in the last layer as a classifier for obtaining labels from the input data as shown in Figure 2.7b. The input layer has a N number of units which is equivalent to the quantity of sample data x. The label layer has C representing y as the number of classes. DDBN trains a joint density model through Discriminative RBM and then each visible label is tested with a test vector. The label which contains the least energy is selected as the best corresponding class. Afterward, we use the backpropagation technique through the entire classifier for fine-tuning the weights for optimal classification.

2.2.4 Convolutional Neural Networks (CNNs)

In classification problems, MLP is not well suitable for some types of data, especially for images. In fact, they are applied to vectors as input data, hence, to apply them to images, we should use hand-designed feature extractors to transform them into vectors. Therefore, this transformation eliminates the spatial information contained in the images, such as forms and edges. The Convolutional Neural Network (CNN) introduced by LeCun *et al.* [119] have revolutioned image processing and overcame the manual feature extraction. CNN is computed directly on images and consists of one or more convolution layers including hyperbolic tangent non-linearities and subsampling layers. The convolutional layers already include non-linearities and, thus, a convolutional layer actually represents two layers. The feature maps of the final subsampling layer are then fed into the actual classifier consisting of an arbitrary number of fully-connected layers. The output layer usually uses softmax activation functions.

Convolutional layers

Convolution is the main building block of CNN. It represents a mathematical operation that merges two sets of information in order to extract different features of the input. The first convolution layer extracts low-level features such as edges, corners, and lines. Whereas, high-level layers extract high-level features like color, shapes, and objects. In our case, the convolution will determine the output of neurons of which are connected to local regions of the input image using a convolution filter. Convolution layer is characterized by an input map I, a bank of kernels K and biases b. Using images, we could have as input an image with height H, width W and RGB channels $\mathrm{C} = 3$ such that $\mathrm{I} \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times\mathrm{C}}$. For a bank of D kernels we have $\mathrm{K} \in \mathbb{R}^{k_1\times k_2\times\mathrm{C}\times\mathrm{D}}$ and biases $b \in \mathbb{R}^{\mathrm{D}}$, one for each kernel.

The output from this convolution procedure is given by:

$$(\mathrm{I} * \mathrm{K})_{i,j} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1}\sum_{c=0}^{\mathrm{C}} \mathrm{K}_{m,n,c} \cdot \mathrm{I}_{i+m,j+n,c} + b \tag{2.44}$$

The convolutional layer uses filters, called also *kernels* which are convolved across the spatial dimensionality of the input in order to produce a 2D activation map. The convolution operation starts from the top-left corner of the input, then each kernel is moved from left to right, one element at a time. Once the top-right corner is reached, the kernel is moved one element in a downward direction, and again the kernel is moved from left to right. This process is repeated until the kernel reaches the bottom-right corner.

As shown in Figure 2.8, the convolution operation is performed by sliding this kernel K over the input I. At every location, the element-wise matrix multiplication and the sum of the result are computed. This sum goes into the feature map. The red area where the convolution operation takes place is called the *receptive field*. Due to the size of the kernel 3×3 the receptive field size is also 3×3. The above example illustrates 2D convolution operation using a 3×3 kernel. Nevertheless, in reality, images are represented as a 3D matrix with dimensions of height, width, and depth, where depth parameter corresponds to red, green, and blue (i.e., RGB) channels. A convolution kernel has a specific height and width, such as 3×3 or 7×7, and by design, it covers the entire depth of its input so it needs to be 3D as well. However, the kernel size becomes $3 \times 3 \times 3$ (note that the depth of the convolution kernel matches the depth of the image, both being 3). The convolutional layer is also able to significantly reduce the complexity of the model through the optimization of its output. The optimization can be performed through three hyperparameters, depth, stride as well as zero-padding.

The depth of the output volume produced by the convolution layers corresponds to the number of kernels that should be used in the model. Reducing this hyperparameter can significantly minimize the number of neurons of the network, but it can also reduce the pattern recognition proficiency of the model. Stride specifies the step with which the convolution kernel moved. By default, the value is 1. Hence, with this value, the receptive field is heavily overlapped and producing extremely large activations. Alternatively, setting the stride to a greater number will provide small feature maps since it is skipping over potential locations. Zero-padding is the simple technique of padding the input volume with zeros around the border. It is important to understand that by using this technique, we maintain the same dimensionality of the input and the output volumes. To compute this, we can make use of the following formula:

$$\frac{(V - R) + 2Z}{S + 1} \tag{2.45}$$

Where V denotes the input volume size (i.e., height ×width×depth), R is the receptive field size, Z is the amount of zero padding set and S represents the stride. If the result from this equation is not equal to a whole integer, we can conclude that the stride has been incorrectly set.

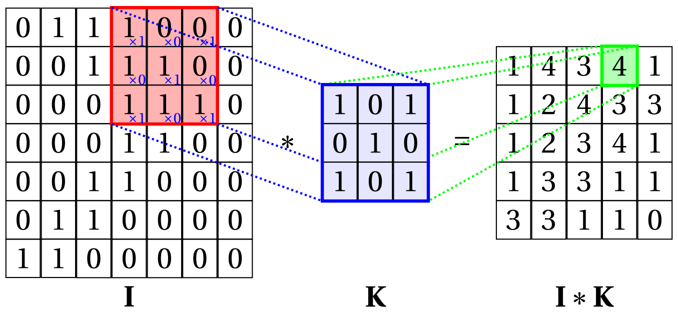


Figure 2.8: 2D convolution using 3 × 3 kernel.

Pooling layers

After a convolution operation, it is common to insert a pooling layer in order to reduce the dimensionality of the representation. Thus further reduce the number of parameters and the computational complexity of the model. As shown in Figure 2.9, the pooling layer operates independently on every activation map in the input and scales its dimensionality (i.e., reducing the height and the width, and the depth is intact) using two ways: max-pooling and average-pooling.

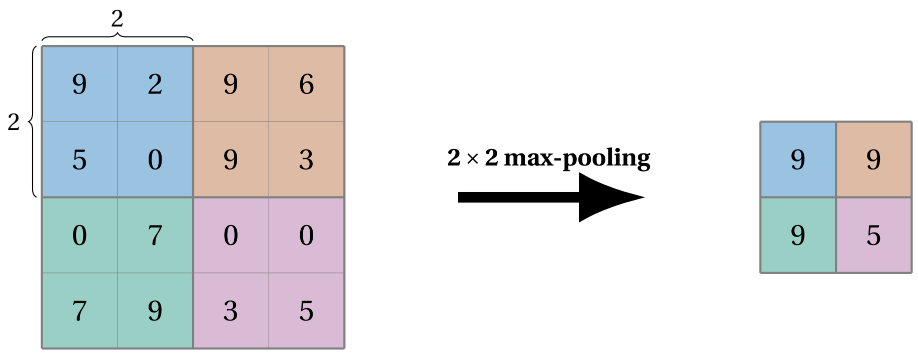


Figure 2.9: Representation of max-pooling with a stride of 2.

In both cases, the input is divided into non-overlapping two-dimensional spaces. Pooling slides a window over its input, and simply takes the max/average value in the window.

Similar to a convolution, the window size and stride are specified. In most CNN architectures, pooling layers are used with 2×2 kernels and applied with a stride of 2 along the spatial dimensions of the input.

Non-linear layers

The Rectified Linear Unit (ReLU) has become very popular in the last few years. It computes the function $f(x) = max(0, x)$ in order to threshold the activation at zero (Figure 2.10). ReLU increases the non-linear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. In comparison to the other non-linear functions used in CNNs (e.g., tanh, and sigmoid), a ReLU shows the advantages that the convergence of stochastic gradient descent is faster, and can be implemented by simply thresholding a matrix of activations at zero.

$$\begin{array}{|cc|cc|} \hline 15 & 20 & -10 & 35 \\ 18 & -110 & 25 & 100 \\ \hline 20 & -15 & 25 & -10 \\ 101 & 75 & 18 & 23 \\ \hline \end{array} \xrightarrow{\textbf{ReLU}} \begin{array}{|cc|cc|} \hline 15 & 20 & 0 & 35 \\ 18 & 0 & 25 & 100 \\ \hline 20 & 0 & 25 & 0 \\ 101 & 75 & 18 & 23 \\ \hline \end{array}$$

Figure 2.10: Representation of ReLU functionality.

Fully-connected layers

After several convolution and pooling layers, the CNN is able to learn a feature hierarchy. To obtain class scores, one or more fully-connected layers are used for classification purposes based on the computed features. These layers perform the same duties found in standard ANN and attempt to produce class scores from the activations, to be used for classification. In a fully-connected layer, all the elements of all the features of the previous layer get used in the calculation of each element of each output feature.

CNN learning

Training a neural network with backpropagation technique consists of two simple phases the feedforward and the backpropagation. In the feedforward phase, a training case is classified using the current neural network. Whereas, in the backpropagation phase a classification error is computed and propagated back through the neural network in order to update the weights/parameters. As CNN is a feedforward neural network with a special structure, its training process is also composed of these two phases: forward propagation and backpropagation (Algorithm 4).

In the forward propagation, the product between each element of the kernel and the input feature map element it overlaps is computed and then the results summed up to obtain the output at that current location. The convolution operation of the input at layer l is given by:

$$x^l_{i,j} = \sum_m \sum_n w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l_{i,j} \tag{2.46}$$

Where l is the l^{th} layer, x is of dimension $\mathrm{H} \times \mathrm{W}$ and has i by j as the iterators, w is kernel of dimension $k_1 \times k_2$ has m by n as the iterators, $w^l_{m,n}$ is the weight matrix connecting neurons of layer l with neurons of layer l-1, and b^l is the bias unit at layer l. After that, we perform ReLU and pooling operations along with forward propagation in the fully-connected layer. At the pooling layer, forward propagation results in an $\mathrm{N} \times \mathrm{N}$ pooling block being reduced to a single value called "winning unit".

In the backpropagation, we compute the gradients of the error with respect to all weights (Equation 2.47) in the network and stochastic gradient descent (Equation 2.48) to update all kernel values/weights and parameter values to minimize the output error.

$$\frac{\partial \mathrm{E}}{\partial w^l_{m',n'}} = \sum_{i=0}^{\mathrm{H}-k_1} \sum_{j=0}^{\mathrm{W}-k_2} \delta^l_{i,j} o^{l-1}_{i+m',j+n'} \tag{2.47}$$

$$w^{t+1} = w^t + \eta \frac{\partial \mathrm{E}}{\partial w^l_{m',n'}} \tag{2.48}$$

Where $\delta^l_{i,j} = \frac{\partial \mathrm{E}}{\partial x^l_{i,j}}$ is the delta matrix which represents all the gradients coming from all the outputs in layer l, and η is the learning rate. In the pooling layer, the error which is acquired by "winning unit" is computed. In the case of max-pooling, this error is just assigned to where it comes from (i.e., winning unit) since the other units in the previous layer's pooling blocks did not contribute to it. In the case of average pooling, the error is multiplied by $\frac{1}{\mathrm{N}\times\mathrm{N}}$ and assigned to the whole pooling block (i.e., all units get this same value).

Algorithm 4 Training in CNN

```
1: Input: training set
2: Initialize all kernels and parameters/weights with random values;
3: Forward propagation step and finding the output probabilities for each class;
4: Calculate the total error at the output layer (according to Equation 2.5);
5: Backpropagation step;
6: Repeat steps 2-4 with all images in the training set.
```

2.2.5 Recurrent Neural Networks (RNNs)

Convolutional Neural Networks are too constrained because they require a fixed-size vector (e.g., images) as input then produce a fixed-sized as output which represents probabilities of different classes. Also, these architectures perform the input and output mapping using a fixed number of layers. In contrary, Recurrent Neural Networks (RNNs) permit to operate over sequences of vectors: sequences in the input (e.g., sentiment analysis [279]), sequences in the output (e.g., image captioning takes an image and outputs a sentence of words [90]), or in the most general case both (e.g., machine translation on which an RNN reads a sentence in English and then outputs a sentence in French). RNNs are a family of neural networks which are developed for discrete sequential data. They are distinguished from the feedforward network by the fact that they use a feedback loop connected to their past decisions. The decision reached at time step $t-1$ affects the decision that will reach at time step t and so on. Hence, RNNs have two sources of input, the present and the recent past which are combined to perform tasks that a feedforward network can not realize. RNNs can be built in many different ways, the most common one used Equation 2.49 or a similar equation in order to define the values of their hidden units. RNNs maintain a latent of hidden state h at time step t that represents the output of a non-linear mapping from their input x^t and the previous state h^{t-1}.

$$h^t = \sigma(wx^t + \mathrm{R}h^{t-1} + b) \tag{2.49}$$

Where w and R are the weight matrixes shared over time, b is the bias, and σ is the activation function. RNNs constitute a very powerful class of computational models that are applied on potential applications such as time series prediction (e.g., financial series), time series production (e.g., motor control), and time classification or labeling (e.g., rhythm detection in music and speech). However, RNNs are limited by the effectiveness of the training procedure applied. Gradient-based methods or "real time recurrent learning" [192, 267] and their combination [209] show an important limitation. The temporal evolution of the path integral over all error signals "flowing back in time" exponentially depends on the magnitude of the weights [84]. This indicates that the backpropagated error quickly either vanishes or blows up. Thus standard RNNs fail to learn in the presence of long time lags between relevant input and target events. One solution to overcome this limitation is Long Short-Term Memory (LSTM).

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTMs) are a special kind of RNN with so-called Long Short-Term Memory units. They were introduced by the German researchers Hochreiter and Schmidhuber [84] as a solution to the vanishing gradient problem. LSTMs work tremendously well on a large variety of problems and are now extensively used. LSTMs consist of units called memory blocks which contain memory cells with self-connections storing the temporal state of the network in addition to special multiplicative units called gates. In these gates, information can be stored in, written to, or read from a cell. Each cell makes decisions about what to store, and when to allow reads, writes, and erasures, via gates that open and close. However, these gates are analog and implemented with element-wise multiplication by sigmoids, which are all in the range of $[0-1]$. This analogy has the advantage over digital of being differentiable, and as a result suitable for backpropagation. The gates act on the signals which they receive. Similarly to the neural network's nodes, they pass or block on information based on its strength and import using their own weights. Those weights, like the weights that modulate input and hidden states, are adjusted via the recurrent network learning process.

LSTM step-by-step

The basic unit in the hidden layer of an LSTM network is the memory block. As shown in Figure 2.11, the memory block consists of one or more memory cells and a pair of adaptive, multiplicative gating units. In summary, each memory block contains:

- memory cell: c_t stores the state;
- forget gate: f_t controls what to forget;
- input gate: i_t controls what to learn;
- output gate: o_t controls the amount of content to modify.

Step 1: the first step relies on deciding what information seems important to keep. The decision is made by the forget layer which looks at x_t and h_{t-1}, then outputs a value between 0 and 1 in the cell state c_{t-1} using a sigmoid function. If the information is important, the forget gate will be closed and keeps them many timesteps, otherwise f_t can reset the memory content.

$$f_t = \sigma(w_f \cdot x_t + u_f \cdot h_{t-1} + b_f) \tag{2.50}$$

Step 2: this step decides the new information that should be stored in the cell state. For that, it provides two main phases. Firstly, the input gate layer i_t decides which values will be updated using the sigmoid function. Secondly, a $tanh$ function creates a vector of new candidates values $\tilde{c}$ that could be added to the state. After that, the above phases are combined to create an update to the state in Step 3.

$$i_t = \sigma(w_i \cdot x_t + u_i \cdot h_{t-1} + b_i) \tag{2.51}$$

$$\tilde{c}_t = tanh(w_c \cdot x_t + u_c \cdot h_{t-1} + b_c) \tag{2.52}$$

Step 3: in this step, the old state c_{t-1} is updated into the new cell state c_t. c_{t-1} is multiplied by the forget gate, this result is added to the new memory content $i_t \times \tilde{c}_t$.

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{2.53}$$

Step 4: finally, the output gate o_t decides the amount of the memory content to yield to the next hidden state. In the first phase, the sigmoid gate controls the part of the cell state that will participate in the output. Then, in the second phase, the cell state is inserted through $tanh$ in order to push the values to be between -1 and 1. Then, the sigmoid output is multiplied with the $tanh$ one.

$$o_t = \sigma(w_o \cdot x_t + u_o \cdot h_{t-1} + b_o) \tag{2.54}$$

$$h_t = o_t \times tanh(c_t) \tag{2.55}$$

Where $(\cdot)$ is the inner products, $w_{(.)}$, and $u_{(.)}$ are the weights, and $b_{(.)}$ is the bias.

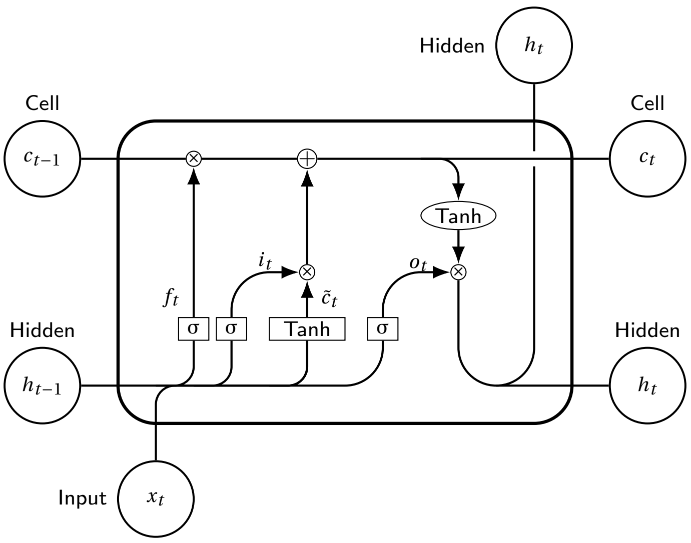


Figure 2.11: LSTM memory block.

LSTM learning

Previously, we have provided Equations 2.50-2.55 which are used in the LSTM forward pass. To perform the backward pass of an LSTM hidden layer within a recurrent neural network, the backpropagation through time algorithm with the exact error gradient is used to train the network. The LSTM equations are given for a single memory block only. For multiple memory blocks, the calculations are repeated for each one.

Forget gate

$$\delta f_t = \delta c_t \odot c_{t-1} \odot f_t \odot (1 - f_t) \tag{2.56}$$

Input gate

$$\delta i_t = \delta c_t \odot \tilde{c}_t \odot i_t \odot (1 - i_t) \tag{2.57}$$

$$\delta \tilde{c}_t = \delta c_t \odot i_t \odot (1 - \tilde{c}_t^2) \tag{2.58}$$

Cell state

$$\delta c_t = \delta h_t \odot o_t \odot (1 - tanh^2(c_t)) + \delta c_{t+1} \odot f_{t+1} \tag{2.59}$$

Output gate

$$\delta o_t = \delta h_t \odot tanh(c_t) \odot o_t \odot (1 - o_t) \tag{2.60}$$

$$\delta h_t = \triangle_t + \triangle_{h_t} \tag{2.61}$$

The final updates to the internal parameters is computed as:

$$\delta \mathrm{W} = \sum_{t=0}^{\mathrm{T}} \delta gate_{s_t} \otimes x_t \tag{2.62}$$

$$\delta \mathrm{U} = \sum_{t=0}^{\mathrm{T}-1} \delta gate_{s_{t+1}} \otimes h_t \tag{2.63}$$

$$\delta b = \sum_{t=0}^{\mathrm{T}} \delta gate_{s_{t+1}} \tag{2.64}$$

Where $\otimes$ is the outer products, $\odot$ is the element-wise product or Hadamard product, $\triangle_t$ is the output difference as computed by any subsequent layers (i.e., the rest of the network), $\triangle_{h_t}$ is the output difference as computed by the next time-step LSTM, and $\triangle_{h_{t-1}} = u^{\mathrm{T}} \cdot \delta gate_{s_t}$.

$$gate_{s_t} = \begin{bmatrix} c_t \\ i_t \\ f_t \\ o_t \end{bmatrix}, w = \begin{bmatrix} w_c \\ w_i \\ w_f \\ w_o \end{bmatrix}, u = \begin{bmatrix} u_c \\ u_i \\ u_f \\ u_o \end{bmatrix}, \text{ and } b = \begin{bmatrix} b_c \\ b_i \\ b_f \\ b_o \end{bmatrix}$$

2.3 Conclusion

In this chapter, we presented a brief history of deep learning, a definition of its concepts, as well as an exhaustive analysis of its techniques and architectures. Firstly, we defined the most basic form of artificial neural networks by providing the analogy between an artificial neuron and biological neuron. Secondly, we introduced deep learning architectures including Restricted Boltzmann Machine (RBM) and its variants, Deep Belief Network (DBN), Convolution Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) as well as their learning process. In summary:

-**deep learning** is a set of techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction, transformation, and classification.

-**Boltzmann Machine (BM)** is a network of symmetrically connected, neuron-like units that make stochastic decisions about whether to be on or off.

-**Restricted Boltzmann Machine (RBM)** is a specific type of BM composed of two layers; visible units and hidden units with no visible-visible or hidden-hidden connections.

-**Deep Belief Network (DBN)** is a probabilistic generative model composed of multiple layers of stochastic, hidden variables.

-**Recurrent Neural Network (RNN)** is a type of advanced artificial neural network (ANN) that involves directed cycles in memory.

-**Long Short-Term Memory (LSTM)** is a special kind of RNN, capable of learning long-term dependencies.

Due to the emergence of deep learning methods in computer vision, all these architectures will be applied in the next chapters in order to perform object classification, scene classification, and topological mapping for mobile robotic.

Part I

Object classification

Chapter 3

Object Classification: literature review

We cannot do anything with an object that has no name.

Maurice Blanchot

Contents

OBJECT recognition is a fundamental task for a large number of computer vision applications, including content-based image retrieval, automated surveillance, and video retrieval. Also, recognizing objects allows Simultaneous Localization And Mapping (SLAM) algorithms for mobile robots to map their environment in order to localize, navigate, and avoid obstacles on it. However, several 2D and 3D object recognition systems have been developed using information acquired from sensors such as RGB and infra-red cameras, lasers, and RGBD cameras. With the arrival of the Microsoft Kinect camera, the depth information can be cheaply obtained besides RGB, which leads to important computer vision applications. Recently, one of the most popular representations of RGB and depth data "point clouds" became useful data representation that supports the classification of 3D objects. In general, 3D object recognition pipelines consist of a dataset of objects which are provided to be recognized in the scene environment. Images are captured for all the objects in the dataset and then different types of information are extracted and stored. Most of these pipelines consider the problem of recognizing objects based on information extracted from a single view. Hence, in real-world situations, a single viewpoint may simply not contain sufficient information to reliably recognize the objects. In this chapter, we provide in the first part a brief discussion on the related work in 2D/3D object classification. Then, we introduce our survey contribution [297] in the context of 3D object recognition.

3.1 State-of-the-art

In the last decades, there has been considerable work in the computer vision field which tackles the challenge of 2D/3D object classification. Here we provide a brief survey of different approaches on object classification.

3.1.1 2D recognition and categorization approaches

Appearance-based object recognition methods use global or local features in order to describe objects. Local appearance methods [98, 260] search for the salient region or points (e.g., corners, entropy, or edges) which characterize the object of interest. Typically, these points are placed at local peaks in a scale-space search and filtered to keep only those that are most likely invariant over transformations. Later, the description of each interest point is built and should be distinctive, concise, and invariant over transformations that are occured by camera pose and lighting changes. These methods are efficient, intensitive to viewpoint changes, and resistant to partial occlusion. Global appearance methods model the information content of the whole object of interest. This information can be represented by contour representations, shape, and texture. Global features are very useful in applications where a rough segmentation of the object is available. Most object recognition methods used either global or local features exclusively. Since it's so difficult to combine a single global feature vector with a set of local features in a suitable way. However, some works [136, 280] showed that combining both local and global features is beneficial for object recognition applications.

Recently, the approaches that were based on Bag of Words (BoWs), also known as Bag of features produced the promising results on several applications, such as object and scene recognition [28, 131], localization and mapping for mobile robots [56], video retrieval [224], text classification [12], and language modeling for image classification and retrieval [130, 155, 288]. Sivic *et al.* [223] used Latent Dirichlet Allocation (LDA) and Probabilistic Latent Semantic Analysis (pLSA) in order to compute latent concepts in images from the cooccurrences of visual words. The authors generated a consistent vocabulary of visual words that is insensitive to viewpoint changes and illumination. For this reason, they used vector quantized SIFT descriptors which are invariant to translation, rotations, and re-scaling of the image. Csurka *et al.* [42] developed a generic visual categorization approach for identifying the object content of natural images. In the first step, their approach detected and described image patches which are clustered with a vector quantization algorithm to generate a vocabulary. The second step constructed a bag of keypoints that counts the number of patches assigned to each cluster. Finally, they used Naive Bayes and Support Vector Machines (SVMs) to determine image categories. Fergus *et al.* [55] suggested an object class recognition method that learns and recognizes object class models from unlabeled and unsegmented cluttered scenes in a scale-invariant manner. The approach exploited a probabilistic model that combined shape, appearance, occlusion and relative scale, as well as an entropy-based feature detector to select regions and their scale within an image. Philbin *et al.* [181] proposed a large-scale object retrieval system with large vocabularies and fast spatial matching. They extracted features from each image in a high-dimensional descriptor space which are quantized or clustered to map every feature to a "visual word". This visual word is used to index the images for the search engine. Wu *et al.* [271] proposed a new scheme to utilize an optimized BoW models called Semantics Preserving Bag of Words (SPBoW) that aims to map semantically related features to the same visual words. SPBoW computed a distance between identical features as a measurement of the semantic gap and tries to learn a codebook by minimizing this gap. Larlus *et al.* [115] combined a BoWs recognition component with spatial regularization based on a random field and a Dirichlet process mixture for category-level object segmentation. The Random Field (RF) component assured short-range spatial contiguity of the segmentation while a Dirichlet process component assures mid-range spatial contiguity by modeling the image as a composition of blobs. Finally, the BoWs component allows strong intra-class imaging variations and appearance. Vigo *et al.* [258] exploited color information in order to improve the BoWs technique. They selected highly informative color-based regions for feature extraction. Then, feature description focused on shape and can be improved with a color description of the local patches. The experiments showed that color information should be used both in the feature detection as well as the feature extraction stages. Khan *et al.* [101] suggested integration of spatial information in the BoWs. The approach modeled the global spatial distribution of visual words that consider the interaction among visual words regardless of their spatial distances. The first step consisted of computing Pair of Identical visual Words (PIW) that save all the pairs of visual words of the same type. The second step represented a spatial distribution of words as a histogram of orientations of the segments formed by PIW. More recently, Hannat *et al.* [72] described an object dataset with SURF feature points which they quantify with the k-means clustering algorithm to make visual words. Then, they trained a SVM classifier having as entries the distribution of the bag of features extracted earlier using GPU implementation. Finally, the results of their experiments showed an average recognition rate between 95% and 100%.

3.1.2 3D recognition and categorization approaches

Most of the recent work on 3D object categorization focused on appearance, shapes, and BoWs extracted from certain viewing point changes of the 3D objects. Savarese and Fei-Fei [207] proposed a compact model for representing and learning 3D object categories. Their model solved scale changes and arbitrary rotation problems using appearance and 3D geometric shape. Each object is considered as a linked set of parts that are composed of many local invariant features. Their approach can classify, localize and infer the scale as well as the pose estimation of objects in the given image. Toldo *et al.* [243] introduced BoWs approach for 3D object categorization. They used spectral clustering to select seed-regions then computed the geometric features of the object sub-parts. Vector quantization is applied to these features in order to obtain BoWs histograms for each mesh. Finally, SVM is used to classify different BoW histograms for 3D objects. Zhong [283] introduced an approach for 3D point cloud recognition based on a new 3D shape descriptor called Intrinsic Shape Signature (ISS). ISS used a view-dependent transform encoding for the viewing geometry to facilitate fast pose estimation, and a view-independent representation of the 3D shape in order to match shape patches from different views directly. Bo *et al.* [26] introduced a set of kernel features for object recognition. The authors developed kernel descriptors on depth maps that model size, depth edges, and 3D shape. The main match kernel framework defined pixel attributes and designed match kernels in order to measure the similarities of image patches to determine low dimensional match kernels. Lai *et al.* [111] built a new RGBD dataset and proposed methods for recognizing RGBD objects. They used SIFT descriptor to extract visual features and Spin Image (SI) descriptor to extract shape features that are used for computing Efficient Match Kernel (EKM). Finally, Linear Support Vector (LiSVM), Gaussian kernel SVM (kSVM) and Random Forest (RF) are trained to recognize both the category and the instance of objects. Mian *et al.* [160] suggested a 3D object retrieval approach from cluttered scenes based on the repeatability and quality of keypoints. The authors proposed a quality measure to select the best keypoints for extracting local features. They also introduced an automatic scale selection method for extracting scale and multi-scale invariant features in order to match objects at different unknown scales. Madry *et al.* [145] proposed the Global Structure Histogram (GSH) to describe the point cloud information. Their approach encoded the structure of local feature response on a coarse global scale to retain low local variations and keep the advantage of global representativeness. GSH can be instantiated in partial object views and trained using complete or incomplete information about an object. Tang *et al.* [236] proposed a Histogram of Oriented Normal Vectors (HONV) feature which is based on local geometric characteristics of an object captured from the depth sensor. They considered that the object category information is presented on its surface. This later is described by the normal vector at each surface point and the local 3D geometry is presented as a local distribution of the normal vector orientation.

Recently, several works proposed real-time approaches for object recognition using 3D sensors. Shin *et al.* [215] presented a new algorithm for mobile robots to learn the concept of objects. Then, they categorized these objects without supervision using 3D point clouds extracted from a laser scanner. Particularly, they addressed the challenges of categorizing objects discovered in different scans without knowing the number of categories. The algorithm found objects per scan and gave them locally-consistent labels. Then, they introduced the class graph that encodes the relationship among local object class labels in order to associate these object labels across all scans. Their algorithm provided a basis for online learning and improved the results of categorization over pure clustering. Boubou *et al.* [32] addressed the problem of robot pursuit based upon a real-time 3D object recognition. They designed a new global online descriptor named Differential Histogram of Normal Vectors (DHONV) to extract the geometric characteristics of the captured 3D surfaces of the objects. DHNOV described the 3D surfaces of an object by quantizing the normalized differential angles of the surface normal vectors into a 1D histogram. The advantage of this online descriptor is the invariance to scale and viewpoint changes. Kasaei *et al.* [97] provided a cognitive architecture to detect objects in crowded scenes and learn new object classes from the set of accumulated experiences in an incremental and open-ended manner. Open-ended implies that the set of object classes to be learned is not known in advance. This means that the training instances are extracted from the online experiences of a robot. Then, they proposed a novel unsupervised Next-Best-View (NBV) prediction algorithm to predict the next best camera pose to improve object detection performance. Tsai *et al.* [252] suggested a new system for simultaneous 3D object recognition and pose estimation in real-world environments. First, the system converted the input RGBD image to colored point cloud data and then computed point cloud segmentation method to obtain object clusters. Before starting the calculation of the feature description, the system extracted features of the scene based on shape information and local texture. Then, it used a PCL descriptor called Color Signature of Histograms of Orientations (CSHOT) to build descriptors of these detected features. A keypoint-based two-stage matching process is performed to speed up the computation of finding correspondences between the object clusters of the current scene and a colored point cloud model. After that, the Hough voting algorithm is utilized in order to filter out matching errors in the correspondence set and estimate the initial 3D pose of the object. Finally, the pose estimation stage employed RANdom SAmple Consensus (RANSAC) and hypothesis verification algorithms to refine the initial pose and filter out poor estimation results with error hypotheses.

3.1.3 Deep learning-based approaches

Nair and Hinton [172] presented a top-level model of DBNs for 3D object recognition. This model is a third-order Boltzmann machine that is trained using a combination of both generative and discriminative gradients. The model performance is evaluated on NORB images where the dimensionality for each stereo-pair image is reduced by using a "foveal" image. The final representation consisted of 8976-dimensional vectors that are learned with a top-level model for DBNs. Socher *et al.* [225] introduced the first convolutional-recursive deep learning model for 3D object recognition. They computed a single CNN layer to extract low-level features from both color and depth images. These representations are provided as input to a set of RNNs with random weights that produce high-quality features. Finally, the concatenation of all the resulting vectors formed the final feature vector for a softmax classifier. Schwarz *et al.* [211] developed a meaningful feature set that results from the pre-trained stage of CNN. The depth and RGB images are processed independently by CNN and the resulting features are then concatenated to determine the category, instance, and pose of the object. Eitel *et al.* [50] presented two separate CNN processing streams for RGBD object recognition. RGB and colorized depth images consisted of five convolutional layers and two fully-connected layers. Both streams are processed separately through several layers and converged into one fully-connected layer and a softmax layer for the classification task. Alex [6] proposed a new approach for RGBD object classification. Four independent CNNs are trained in a sequence, one for each depth data and three for RGB data. The decisions of each network are combined to obtain the final classification result. Maturana and Scherer [154] proposed a new 3D CNN architecture for accurate object detection from LiDAR and RGBD point clouds. They integrated a volumetric occupancy grid representation which estimated the spatial occupancy with a supervised 3D CNN that predicted a class label directly from the occupancy grid. Finally, they evaluated VoxNet on publicly available benchmarks using LiDAR, RGBD, and CAD data and showed that it achieved accuracy beyond the state-of-the-art performing classification in real-time. Ouadiay *et al.* [179] proposed a new approach for real 3D object recognition and categorization using DBNs. First, they extracted 3D keypoints from point clouds using 3D SIFT detector and then they computed SHOT/SHOTCOLOR descriptors. The performance of the approach is evaluated on two datasets: Washington RGBD object dataset and real 3D object dataset. Madai *et al.* [144] reinvestigated Deep CNNs (DCNNs) for RGBD object recognition. They proposed a new method for depth colorization based on surface normals, which colorized the surface normals for every pixel and computed the gradients in a horizontal direction (x-axis) and vertical direction (y-axis) through the Sobel operator. The authors defined two 3D vectors a and b in direction of the z-axis in order to calculate the surface normal n. As n has 3 dimensions, the authors mapped each of the three values of the surface normal to a corresponding RGB channel. Hedge *et al.* [77] proposed a method for 3D CAD model classification which used two data representations: volumetric representation, and pixel representation. In volumetric representation, the 3D object is discretized spatially as binary voxels with value -1 if the voxel is occupied and 0 otherwise. While, in pixel representation, the 3D object is represented as a set of projected 2D pixel images. The authors introduced two new convolutional networks for volumetric data V-CNN I and V-CNN II that perform well on the partially non-overlapping set of objects which they believed stems from representation dependent features learned from the two representations. Also, they combined the two networks to improve the classification performance and discussed the probable reasons for such a significant boost in performance. Qi *et al.* [186] suggested a novel deep neural network named PointNet that directly takes point clouds as input. PointNet provided a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing. It generated several outputs either class labels for the entire input or per point segment/part labels for each point of the input. Finally, the authors provided an empirical and theoretical analysis on the stability and efficiency of their network. Gomez-Donoso *et al.* [66] proposed LonchaNet, a sliced-based CNN architecture for point cloud classification. First, for every example in the dataset, the approach generated three slices of the input point cloud (i.e., one per 3D axis), then projected the points to a plane, thus generating three images from every example. Each of these images that shape a single example is then fed to a deep CNN. The authors introduced also their second contribution that relies on using the existing GoogLeNet network. They learned specific features of each cross-section or slice from the 3D model with three independent GoogLeNet networks which are concatenated and fed to a fully-connected layer. LonchaNet outperformed most existing approaches that participated in the ModelNet challenge with a success rate of 94.37% in the ModelNet-10 accuracy test thanks to learning discriminative representation. Zhi *et al.* [282] designed a lightweight 3D CNN (LightNet) for real-time 3D object recognition. LightNet consisted of a small number of training parameters as compared to the existing models including VoxNet, FusionNet, and VRN Ensemble. It learned effectively 3D representations using multitask learning, including category and orientation prediction from both entire and partial shapes. LightNet achieved nearly the state-of-the-art recognition accuracy on the ModelNet and Sydney Urban Objects datasets. Experimental results showed that the model improved the VoxNet model by relative 17.4% and 23.1% on the ModelNet10 and ModelNet40 benchmarks with less than 67% of training parameters. Bobkov et *al.* [27] suggested a point pair descriptor that is robust to noise and occlusion. It achieved high accuracy in object retrieval and classification and can be used in a 4D CNN for the task of object classification. The authors proposed also a novel 4D convolutional layer that is able to learn class-specific clusters in the descriptor histograms. 4D CNN outperformed existing deep learning approaches on three benchmark datasets. Loghmani *et al.* [140] introduced a new end to end architecture for RGBD object recognition called recurrent convolutional fusion (RCFusion). In the first step, the method used two streams of CNNs, with the same architecture to process both RGB and depth data (RGB-CNN and Depth-CNN), respectively, and extract features at different network levels. Then, these features are individually transformed through projection blocks and then concatenated to generate the corresponding RGBD features which are fed to an RNN that produces descriptive and compact multimodal features. In the RNN output, a softmax classifier is used to infer the object label. The architecture is trained end to end using backpropagation algorithm based on stochastic gradient descent. Also, their method formulated a loss function that promoted orthogonality between corresponding RGB and depth features to learn complementary information. Finally, the authors conducted extensive experiments on RGBD object dataset and JHUIT-50 then showed that the combination of complementary features representing different levels of abstraction, produced a highly discriminative description of the RGBD data. Sun *et al.* [230] designed a PCA–CCA network method for RGBD object recognition. It is composed of PCA filter layer, CCA filter layer, binary hashing, and block-wise histograms. In the first layer, Principal Component Analysis filters (PCA filters) are learned separately for RGB and depth in order to extract the most discriminative features in both modalities. Then, in the second layer, the CCA method generated the filters for the RGB and depth components, by maximizing the correlation between the two projected sets of variables. In this way, different characteristics of the RGB and depth modalities and the correlation between them are considered by the network. Compared with CNN-based methods, PCA–CCA contains few stages of convolution and few parameters to be fine-tuned which make it efficient even without graphics processing unit acceleration. Experiments were achieved on the Washington RGBD object dataset and demonstrated that PCA-CCA obtained an accuracy that is comparable to state-of-the-art methods.

3.2 Evaluation of PCL's descriptors

With the advent of new 3D sensors like Microsoft Kinect, 3D perception became fundamental vision research in mobile robotic applications like scene manipulation or grasping, scene understanding, and 3D point cloud classification. The Point Cloud Library (PCL) was developed by Rusu *et al.* [202] in 2010 and was officially published in 2011. This open source library, licensed under Berkeley Software Distribution (BSD) terms, represents a collection of state-of-the-art algorithms and tools that operate with 3D point clouds to solve common problems such as object recognition, registration of point clouds, segmentation, and surface reconstruction. Several studies have been made based on PCL's detectors and descriptors, allowing for 3D object recognition applications. Alexandre [5] presented an overview of the state-of-the-art 3D features for both object and category recognition. The approach focused on using a single recognition pipeline that illustrated the available descriptor algorithms in version 1.6 of PCL. Tamas and Jensen [235] analyzed the characteristics of the feature descriptors in terms of robustness against typical disturbances in the context of the object recognition for depth data with intensity information. Their approach used time-of-flight camera and contained two phases: an off-line training for extracting and storing the characteristics of the object, and online testing in which the extracted features of an object is searched within the dataset constructed in the previous phase. Alhamzi *et al.* [7] proposed a system that used a hybrid technique based on Viewpoint Feature Histogram (VFH) and Fast Point Feature Histogram (FPFH) methods. VFH is used as a global descriptor to recognize the object, while FPFH is used as a local descriptor to estimate the position of the object in the real-world scene. More recently, Martínez-Gómez *et al.* [151] proposed a new framework for semantic localization implemented in the PCL library. They generated global descriptors from local descriptors using the BoWs method. Then, they evaluated the framework with different detectors and descriptors.

In this chapter, we use the object model acquisition with markers of RGBDemo software in order to create our own dataset. Then, we evaluate the existing feature extraction methods stemming from a public PCL library in the context of the object recognition. And, for this purpose, we suggest a new recognition pipeline of 3D point clouds based on PCL's descriptors and the recognition threshold that determines the right matching between the point cloud dataset. The main contributions are:

- acquiring 3D object model using RGBDemo as well as the rotating support;
- extracting objects from 3D scenes;
- evaluating the most existing 3D descriptors in version 1.7 of PCL;
- computing a recognition threshold for each class of objects (i.e., point clouds) in order to improve the classification rate.

3.2.1 Point clouds

The point cloud is a data structure which represents three-dimensional data. In the 3D cloud, the points are usually described by their x, y and z geometric coordinates of a sampled surface. When the point cloud contains the color information, the structure becomes four-dimensional data. Point clouds can be obtained using stereo cameras, 3D laser scanners, time-of-flight cameras or Kinect.

In 3D space, points are defined in a clockwise reference frame that is centered at the intersection of the optical axis with the plane which contains the front wall of the camera. The reference frame is decomposed as follows:

- x-axis: is horizontal and directed to the left;
- y-axis: is vertical and faces up;
- z-axis: coincides with the optical axis of the camera. It is turned towards the object.

3.2.2 RGBD camera

In general, the basic version of Microsoft Kinect camera is composed of the color camera, IR emitter, IR depth sensor, a multi-array microphone, and a motorized tilt (Figure 3.1). The depth sensing works with the principle of structured light and combines the IR emitter and the IR depth sensor. The distance between objects and the camera is ranging from $1.2m$ to $3.5m$. Here, color camera is able to provide the image with the resolution of 640×480 pixels at 30Hz. It also has the option to produce higher resolution images (1280×1024 pixels), running at 10Hz. Kinect's 3D depth sensor (i.e., IR emitter and IR depth sensor) similarly to the color camera, can provide depth images with the resolution of 640×480 pixels at 30Hz.

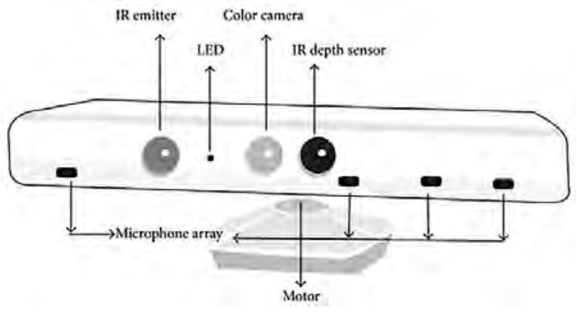


Figure 3.1: The structure and internal components of the Kinect camera.

3.2.3 RGBD datasets

Washington RGBD dataset

Washington RGBD dataset is a large dataset built for 3D object recognition and categorization applications. It is a collection of 300 common household objects which are organized into 51 categories. Each object is placed on a turntable and is captured for one whole rotation in order to obtain all object views using the Kinect camera that records synchronized and aligned 640 × 480 RGB and depth images at 30 Hz [111]. Figure 3.2 shows some example objects from the dataset. Each shown object belongs to one of the 51 object categories. Although the background is visible in these images and will be segmented using segmentation masks. The segmentation relies on removing most of the background by taking only the points within the turntable and object. Then, RANSAC plane fitting is performed to find the table plane and take points that lie above it to be the object. Figure 3.3 shows different types of RGBD Washington data that include RGB, depth, and point cloud extensions.

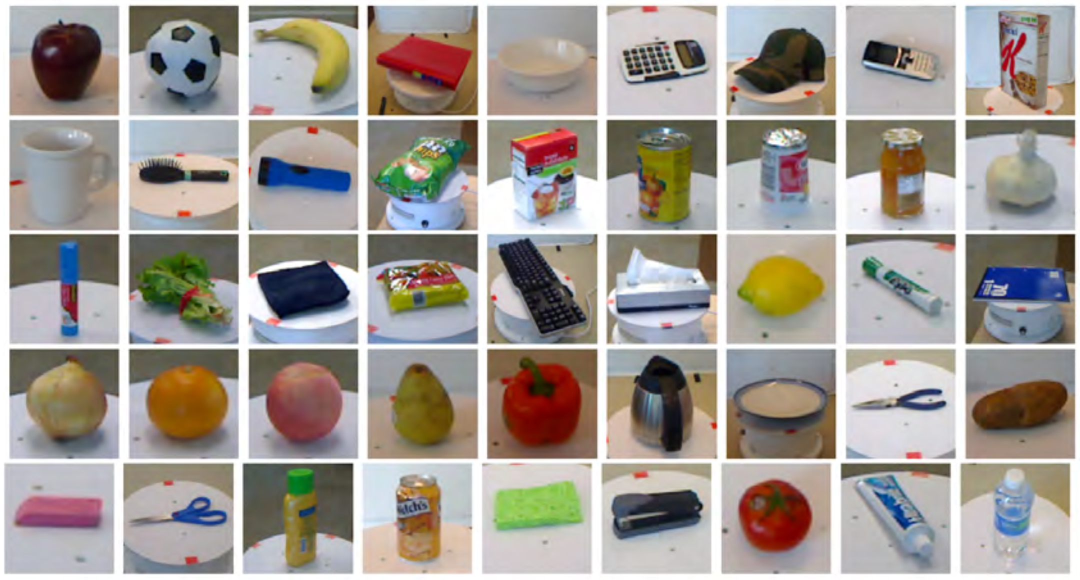


Figure 3.2: Objects from the RGBD Object Dataset. Each object shown here belongs to a different category [111].

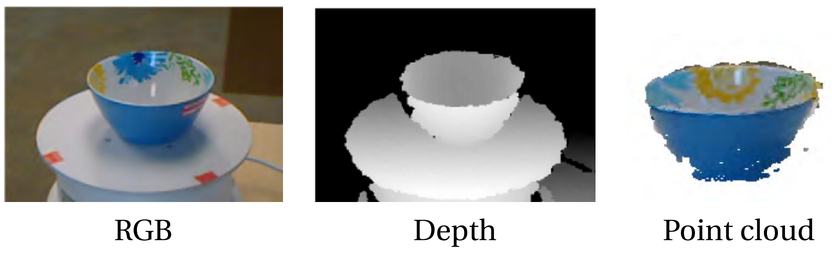


Figure 3.3: Different types of RGBD Washington object dataset.

Our RGBD dataset

Here, we present in detail the acquisition of 3D object models. It aims to gather and represent the information associated with a real-world object using multiple views as captured by Kinect.

Software RGBDemo is an open source software that provides a simple toolkit to start fusion with Kinect data and develop standalone computer vision programs. The project consists of a library called nestk, which is designed to easily integrate into existing cmake-based software and provides quick access to the Kinect features. It includes OpenCV for image processing, QT for the graphical parts, libfreenect for Kinect, and PCL library. The RGBDemo software contains several demonstrations using the Kinect for camera calibration, scene reconstruction with SLAM, people detection and localization, and object model acquisition with markers. The main idea of the last demonstration is to build a 3D model for real-world objects using open source Aruco library (BSD licensed), which is able to generate and recognize square markers and issue the ID and coordinates of the corners of each detected marker. Fiducial markers are presented in the form of black squares with a binary pattern inside and are commonly used in augmented reality applications to robustly track a plane.

Rotating support The main purpose is to build rotating support while keeping the Kinect camera fixed. This rotating support consists of four fiducial markers aligned at fixed positions to form a rectangular shape. The board is actuated by a precise motor so that we know the pose of the object at each Kinect frame. As shown in Figure 3.4, we use a stepper motor to achieve very precise positioning and speed control. For the precision motion, the stepper motor is maintained at 5 volts from the serial port of the computer using Arduino Kit and Adafruit Motor Shield.

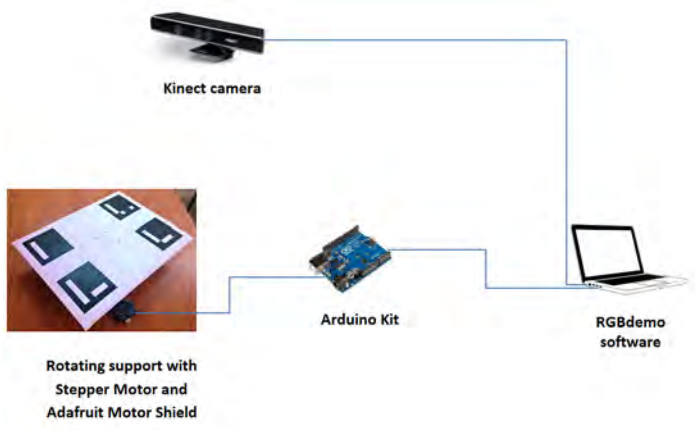

Kinect camera
Arduino Kit
RGBdemo software
Rotating support with Stepper Motor and Adafruit Motor Shield

Figure 3.4: 3D object model acquisition interface.

Setup The easiest way to build such support is to print the markers on two sheets of paper and glue them on the support. This leaves enough empty space in the center to put our object without occluding the markers. Four markers allow a robust estimation of the Kinect pose in each frame since only three markers have to be successfully detected. To generate the viewpoints, the Kinect's position needs to be fixed during the movement of the support in order to ensure a constant illumination and avoid the risk of having desynchronized depth and color images.

3.2.4 Overview of our 3D recognition pipeline

Ideally, a 3D recognition pipeline (Figure 3.5) should be able to grab 3D scenes from the Kinect camera, segment them to extract objects, compute local or global descriptors for each

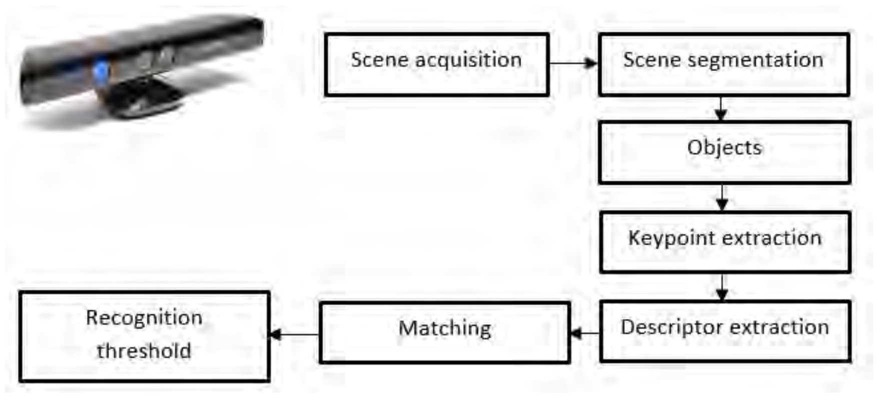


Figure 3.5: Our 3D object recognition pipeline.

point cloud, compare them with the ones stored in our dataset, and output all matches with their distance metrics.

- **Scene acquisition**: in Figure 3.6, we use Kinect camera to capture point clouds of our 3D scenes.

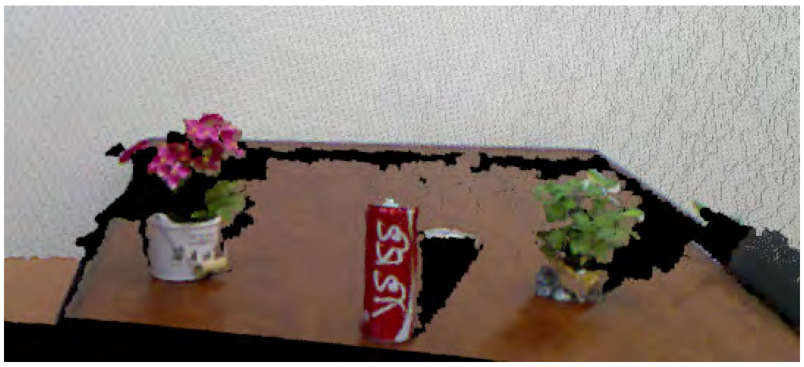


Figure 3.6: Our 3D scene acquisition.

- **Scene segmentation**: 3D scenes contain several objects which should be separated from the background. For this reason, we use a segmentation technique to keep only point clouds of objects.

- **Keypoint extraction**: the first step of 3D perception pipeline is represented by the extraction of 3D keypoints (i.e., interest points) from data. They reduce the computational complexity by identifying particularly those regions of 3D point clouds, which are important for descriptors, in terms of high information density.

- **Descriptor extraction**: once keypoints are extracted, descriptors are computed on the obtained keypoints and these form a description that is used to represent the input cloud.

- **Matching**: descriptors are then matched in order to compare two point clouds. We need a method to match the input cloud with all the ones learned in the training set. For this comparison, different distance metrics can be used. The smallest distance should indicate the input cloud and the most similar one from a dataset.

- **Recognition threshold**: as a result of the matching stage, our pipeline determines the most similar input clouds. However, in some cases, the smallest distance can create a false recognition for objects. To handle this problem, we suggest a matching threshold to reject a false recognition.

3.2.5 Scene segmentation

After the acquisition step, the object must be segmented to extract it from the scene. There are many techniques that perform this task. In our case of study, we use SAmpleConsensus Segmentation (SAC Segmentation) method implemented in the segmentation module of the PCL library. It represents a simple and effective technique for segmenting a point cloud into distinct clusters. SAC segmentation is based on extracting indices filter to finding the dominant plan and subtract it. This plane fitting is often applied to detect common indoor surfaces, such as floors, table tops, and walls. Thereafter, the Euclidean cluster extraction method is used in order to group each object cloud into clusters.

3.2.6 Descriptor matching

The PCL's descriptors are applied for each input cloud and all the ones stored in the dataset. The descriptors are compared using the L2-distance. In Equation 3.1, a similarity measurement is exhaustively computed involving the L2-distance between two point clouds. Equation 3.1 shows also that the two-point clouds are similar when the distance is minimal.

$$\mathrm{L2}(\mathrm{IC}, \mathrm{D}ataset_{(j)}) = \sum_{n=1}^{size} (desc_n(\mathrm{IC}) - desc_n(\mathrm{D}ataset_{(j)}))^2 \tag{3.1}$$

- $size$ = presents the size of PCL's descriptors, it depends on the type of the descriptor;
- $desc_n$= presents the descriptor;
- IC= Input Cloud (i.e., segmented object);
- $\mathrm{D}ataset_{(j)}$: j^{th} point cloud of the dataset.

3.2.7 Recognition threshold

The matching test causes sometimes a false recognition, especially when the input cloud is a negative sample. The system returns the minimum distance between the input cloud and the ones contained in the dataset, despite the fact that this point cloud does not exist. For this reason, we devised a method that would allow our pipeline to reject false recognition. It is essentially based on a statistical calculation of the threshold of a point cloud belonging to a specific class of objects (i). Considering a set of point clouds of a specific class of objects (i), the threshold of each one of those point clouds is defined from the maximum L2-distance between the descriptor of each point cloud and the average of all the descriptors of this set of point clouds. The input cloud belongs to the class (i) when the minimal distance is less than the recognition threshold of the specific class (i). Equation 3.2 shows the calculation of the threshold for each class of objects (i).

$$\tau_i = max(\mathrm{L2}(\mathrm{D}escriptors_{(i)}, mean_{(i)})) \tag{3.2}$$

- $\mathrm{D}escriptors_{(i)}$ = matrix of descriptors for all point clouds stored in the class of objects (i);
- $mean_{(i)}$ = mean of the Descriptors matrix for the class of objects (i).

3.2.8 Evaluating 3D descriptors and detector

Local descriptors

Local descriptors describe individual points that we give as input and focus on the local geometry around that point. The local descriptors are developed for specific applications such as registration, local surface categorization and object recognition. Table 3.1 summarizes the characteristics of PCL's descriptors.

Point Feature Histograms (PFH) PFH [200] was developed to estimate both the surface normals and the curvature. In the first step, the algorithm pairs each point p with all points in its neighborhood denoted qk, and pairs also the neighbors with themselves. Then for each pair, a fixed reference frame consisting of the three unit vectors (u, v, w) is built centered on p using the following procedure:

$$\begin{aligned} u = n_s &: vector\ of\ the\ surface\ normal\ at\ p \\ v &= u \times \frac{(p-q)}{d} \\ w &= u \times v \end{aligned} \tag{3.3}$$

With: $d = \| p - q \|^2$
The difference between the normals at p and q can be represented by:

$$\begin{aligned} \alpha &= \arccos(v.n_q) \\ \phi &= \arccos(u.\frac{p-d}{d}) \\ \theta &= \arctan(w.n_p, u.n_q) \end{aligned} \tag{3.4}$$

For the pairs (p, q), the difference of normals is computed and described with 3 angles (α, ϕ, θ) around the axis, and the distance is ignored since it is variant with the viewpoint. This description is binned into a 125-bin histogram by considering that each of angles can fall into 5 distinct bins, and the final histogram encodes in each bin a unique combination of the distinct values for each angle. Since the PFH descriptor uses all possible point pairs of the k neighbors of p, it has a complexity of $\mathcal{O}(nk^2)$ for a point cloud with n points. There is also a version of PFH that includes color information: PFHRGB. This variant includes three more histograms, one for the ratio between each color channel of p and the channel of q. These histograms are binned as the 3 angles of PFH and hence produce another 125 float values, giving the total size of 250 values for PFHRGB.

Fast Point Feature Histograms (FPFH) FPFH [199] is an amelioration of the PFH descriptor which allows reducing the complexity of calculation from $\mathcal{O}(nk^2)$ to $\mathcal{O}(nk)$, and the time calculation at the expense of precision. In the first step, FPFH considers only the direct connections between the current point p and its k neighbors, removing additional links between neighbors. Then, it computes the histogram of the three angles in the same way as in PFH descriptor. This produces the Simplified Point Feature Histogram (SPFH). In the second step, for each point p, the values of the SPFH of its k neighbors are weighted by their spatial distance $\omega_i = d$ in order to generate the FPFH at p, where FPFH(p) = SPFH(p) + $1/k\Sigma_{i=1}^{k}$SPFH$(i)/\omega_i$. Finally, the three angles are binned in 11-bin histograms and concatenated for each point into a single descriptor 33-bin FPFH.

Signature of Histograms of OrienTations (SHOT) Proposed by Tombari *et al.* [245], SHOT is a descriptor based on the histograms of normal. It is established from the intersection between signatures and histograms, so as to achieve a better balance between the descriptive character and the robustness. In addition, it presents the descriptive power of the 3D shape of the surface that was repeatable and robust to noise, translations, and rotations. It represents an enormous gain in computing time. The description of the geometrical information about the point positions contained in support is made by a set of local 3D histograms defined on a 3D spherical grid that partitions the space according to the radial axes, azimuth, and elevation. For each sector of the grid, the values of the cosine of angles between the normal reference and all their neighbors are accumulated to form the normal histogram with 32 bins. The estimation of the normal is made by calculating a new covariance matrix as a linear combination of the distances of the points belonging to spherical support of the keypoints. The eigenvectors of this matrix from orthogonal directions are repeatable and robust to noise. It is possible to improve the discriminating power of the descriptor by introducing geometrical information concerning the location of points inside the support, in order to obtain a signature. It makes by calculating a first set of local histograms on 3D volumes defined by a 3D grid overlaid on the support and then grouping all local histograms to form the resulting descriptor. More recently, CSHOT (SHOTCOLOR) version [246] combines SHOT information on the shape, texture, and colors. This descriptor is a combination of a normal histogram and a color one. The color histogram is formed by RGB absolute values between the reference point and their neighboring ones.

Spin Image (SI) SI [91] was originally designed to describe surfaces made by edges, vertices, and polygons, but it has been since adapted for point clouds. This descriptor translates the local properties of the surface oriented in a coordinate system fixed and linked to the object. This system is independent to the viewing changes. The spin is defined at a point oriented and designated by its 3D position (p) as well as associated direction (n the normal to the local surface). A 2D local coordinate base is formed using the tangent plane P in the point p, oriented perpendicularly to the normal n, and the line L through p parallel to n. A cylindrical coordinate system (α, β) of the point p is then deduced. The radial coordinate defining the distance (non-negative) is perpendicular to L and the elevation coordinate of the defined distance is perpendicular to P (signed positive or negative). The resulting histogram is formed by counting the occurrences of different pairs of discretized distances.

Global descriptors

Global descriptors describe object geometry. They are not computed for individual points, but for a whole cluster. They are high-dimensional representations of object geometry. Global descriptors are more efficient in object recognition, geometric categorization, and shape retrieval. They are usually calculated for subsets of the point clouds that are likely to be objects.

Viewpoint Feature Histogram (VFH) VFH [201] is a global descriptor that produces only one descriptor for the input cloud, that adds viewpoint variance to the FPFH by using the viewpoint vector direction. The VFH consists of two components: a viewpoint direction component, and the component associated to FPFH. The procedure used for the description is as follows:

1- calculate the centroid c of the point cloud and its normal n_c;

2- for each point p in the cloud, build the local reference frame (u, v, w) using $(u = n_c)$, $(v = (p - c) \times u)$, and $(w = u \times v)$.

3- find the angles (α, ϕ, θ) using this reference frame as in the PFH.

The angle $\beta = \arccos(n_p.c / \parallel c \parallel)$ presents the central viewpoint direction translated to each normal makes with each point's normal. The three angles (α, ϕ, θ) with the distance d between each point and the centroid are binned into a 45-bin histogram, also the β is encoded in a 128-bin histogram. The total length of the VFH descriptor is the combination of these two histograms and is equal to 308 bins. Finally, the bins are normalized using the total number of points in the point cloud. This makes the VFH descriptor invariant to scale.

Clustered Viewpoint Feature Histogram (CVFH) CVFH [4] is an extension to VFH descriptor in order to estimate a more robust coordinate frame that could handle the different data properties of the models and scenes. The basic idea is that objects have a certain structure that allows to split them into a certain number N of disjoint smooth regions. Each of these smooth regions is then used independently to compute a set of VFH histograms. In the first step, CVFH segments the point cloud into clusters (i.e., stable regions) of neighboring points with similar normals. The smooth regions for CVFH are easily computed by removing points on the object with high curvatures that indicate noise or borders between smooth regions. It performs region growing on the remaining points on XYZ and normal space then calculates the VFH for each cluster. Finally, it adds a shape distribution quotient to each histogram that expresses how the points are distributed around the centroid. Each cluster C_i is defined by the pair (c_i, n_i) (i.e., center of gravity and the normal). Then it independently deployed as one of the axes of a pointwise reference frame from which three angular distributions (each made out of 45 bins) are computed. CVFH includes fourth and fifth components (i.e., 45 and 128 bins respectively) into the histogram, the fourth being based on the L1-distribution obtained from C_i. The fifth is obtained from another angular distribution for each n_j and the central view direction. The total size of a CVFH histogram is 308.

Oriented, Unique and Repeatable CVFH (OUR-CVFH) OUR-CVFH [3] expands the previous descriptor CVFH. It adds a computation of a unique reference frame to make it more robust. OUR-CVFH relies on the use of Semi Global Unique Reference Frames (SGURFs), which are repeatable coordinate systems computed for each region. The first part of the computation is similar to CVFH, but after segmentation, the points in each region are filtered once more according to the difference between the region's average normal and their normals. This result in best-shaped regions improves the estimation of the Reference Frames (RFs). After that, the SGURF is computed for each region. Disambiguation is performed to determine the signs of the axes, according to the point distribution. If this is not enough and the sign remains ambiguous, multiple RFs will need to be created to account for it. Finally, the OUR-CVFH descriptor is computed.

Ensemble of Shape Functions (ESF) ESF [268] is an ensemble of ten 64-bin-sized histograms resulting in a single 640 values histogram for a given input point cloud of shape functions describing a characteristic of the point cloud. ESF uses a voxel grid as an approximation of the real surface. It iterates through all the points in the cloud using for every iteration 3 random points. For these points, the shape functions are computed:

- D2: computes the distances between point pairs. For every pair, it checks if the line that connects both points lies entirely inside the surface, entirely outside, or both. Depending on this, the distance value will be binned to one of three possible histograms(i.e., in, out or mixed).

- D2 ratio: is the ratio between parts of the line inside the surface and parts outside. This value will be 0 if the line is completely outside, 1 if completely inside, and some value in between if mixed.

- D3: computes the square root of the area of the triangle formed by the 3 points. Like D2, the result is also in, out or mixed, each with its own histogram.

- A3: computes the angle formed by the points. Then, the value is binned depending on how the line opposite to the angle is (i.e., in, out or mixed).

The ESF descriptor can be efficiently calculated directly from the point cloud with no necessary pre-processing, such as smoothing, hole filling, or surface normal calculation. It handles gracefully data errors such as outliers, holes, noise, and coarse object boundaries. The ESF algorithm differs from the other feature algorithms as it does not require the use of normals to describe the cloud.

Descriptor	**Type**	**Method**	**Input**	**Size**
PFH	L	Hi	Points + Normals + Search method + Radius	125
PFHRGB	L	Hi	Points + Normals + Search method + Radius+Color	250
FPFH	L	Hi	Points + Normals + Search method + Radius	33
SHOT	L	Hy	Points + Normals + Radius	352
CSHOT	L	Hy	Points + Normals + Radius+Color	1344
SPIN	L	Hi	Points + Normals + Radius + Image resolution	153
VFH	G	Hi	Points + Normals + Search method + Radius	308
CVFH	G	Hi	Points + Normals + Search method + Angle + Curvature	308
OUR-CVFH	G	Hi	Points + Normals + Search method + Angle + Curvature	308
ESF	G	Hi	Points + Normals + Radius	640

Table 3.1: General parameters for PCL's descriptors. (∗ G: Global, ∗ L: Local, ∗ Hi: Histogram, ∗ Hy: Hybrid.)

3D Scale-Invariant Feature Transform Detector (3D SIFT)

The algorithm consists of the detected feature points of an image used to characterize every point that needs to be recognized by comparing its characteristics with those of the points contained in other images. The general idea of SIFT is to find the keypoints that are invariant to several transformations/changes: rotation, scale, illumination and viewing angle. The 3D SIFT detector [212] use the Difference-of-Gaussian (DoG) function to extract the extrema points in both spatial and scale dimensions. The scale space of a 3D input point cloud is defined as a function with four dimensions (4D) $L(x, y, z, \sigma) = G(x, y, z, k\sigma) * P(x, y, z)$ that was generated by convolving a 3D variable-scale centered Gaussian kernel $G(x, y, z, \sigma)$ with input point $P(x, y, z)$, where:

$$G(x, y, z, k\sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^3} e^{-\frac{(x^2+y^2+z^2)}{2\sigma^2}} \tag{3.5}$$

σ represents a scale space which is separated by k a constant multiplicative factor. The candidate keypoints in 4D scale space are selected as the local extrema (i.e., minima or maxima) of the multi-scale DoG defined as $D(x, y, z, k^i\sigma) = L(x, y, z, k^{i+1}\sigma) - L(x, y, z, k^i\sigma)$. Then, each sample point is compared to its 80 neighbors (i.e., $27 + 26 + 27 = 80$, 26 neighbors belong to the current scale, and every 27 neighbors in the scale below and above) in order to find extrema of the multi-scale DoG function. A keypoint is selected only if it is smaller than all of its neighbors or larger than all of them. After that, 3D SIFT eliminates the bad candidate keypoints having low contrast by using a thresholding method. A contrast threshold is applied on $D(x, y, z, k^i\sigma)$ to eliminate all the candidate keypoints which are below a fixed threshold value τ.

3.2.9 Experimental results

We compare the quantitative as well as the qualitative performance of the most existing PCL's descriptors. For the keypoint extraction, we have used 3D SIFT detector with both global and local descriptors. 3D SIFT operates on 3D Gaussian filters with increasingly large scales to the voxelized models. In our experiments, we used the following parameters: min scale= 0.001f, number of octaves=2, number of scales per octave=1, and min contrast= 0.001f. To make a comparison, all pipeline stages were treated fairly, using the same parameters and platform.

RGBD dataset

The intention is to build up a large dataset of 3D indoor environment objects for robotic applications. After dataset acquisition using RGBDemo and rotating support, we obtained meshes that represent 3D object models. To improve the quality of these acquired data, we used Meshlab[1] software that was developed at the Visual Computing Lab. It implements a wide range of algorithms and filters that enhance the 3D model reconstruction. The final meshes were converted to point clouds data using the PCL tools.

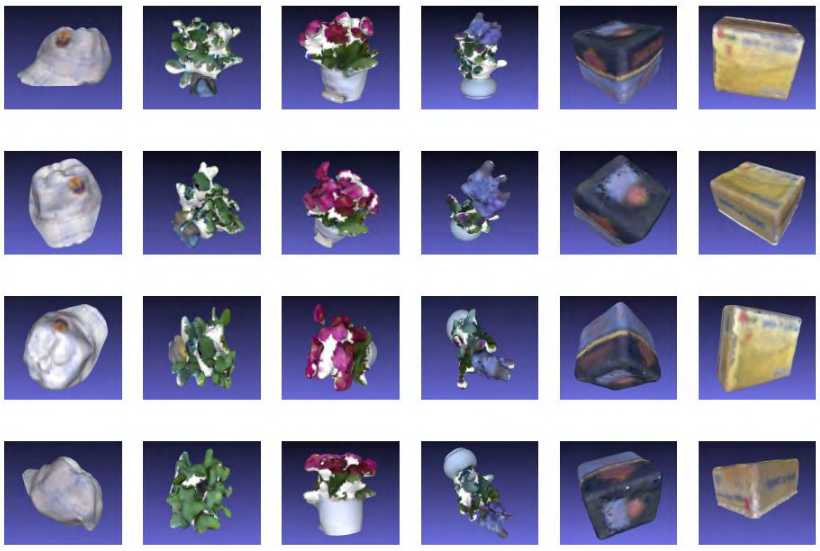


Figure 3.7: The sample objects from our training object classes.

The training dataset is illustrated in Figure 3.7. It contains point clouds for each object captured several times, that represent 6 classes: cap1, flour1, flour2, flour3, tea and box. After the 3D scene acquisition, we segment all the objects which are present in it. Each object is saved as the point cloud and stored in the testing dataset. Subsequently, we will use the segmented point clouds to test our 3D recognition pipeline which is based on the training dataset recognition thresholds. To validate the performance of our dataset and 3D recognition pipeline, we tested our method on Washington RGBD dataset. For that, we selected 400

[1]http://meshlab.sourceforge.net/

point cloud data as a training dataset (Figure 3.8a). These point clouds are captured from different camera views representing 4 classes that look like objects in our dataset. For testing dataset (Figure 3.8b), we selected 100 point clouds which contain objects from the 4 classes as well as negative samples that are not belonging to the training set.

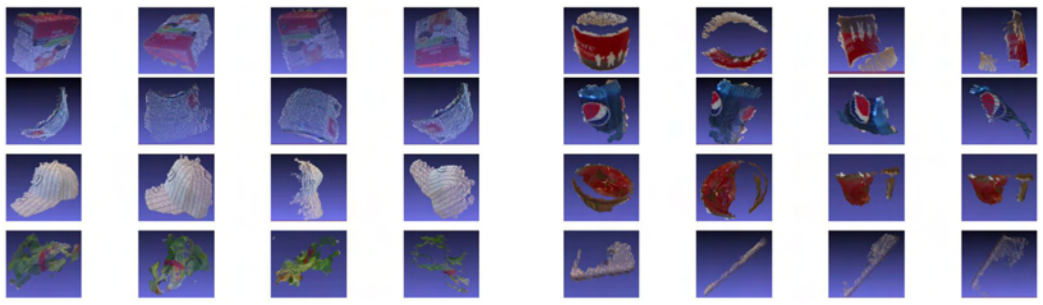
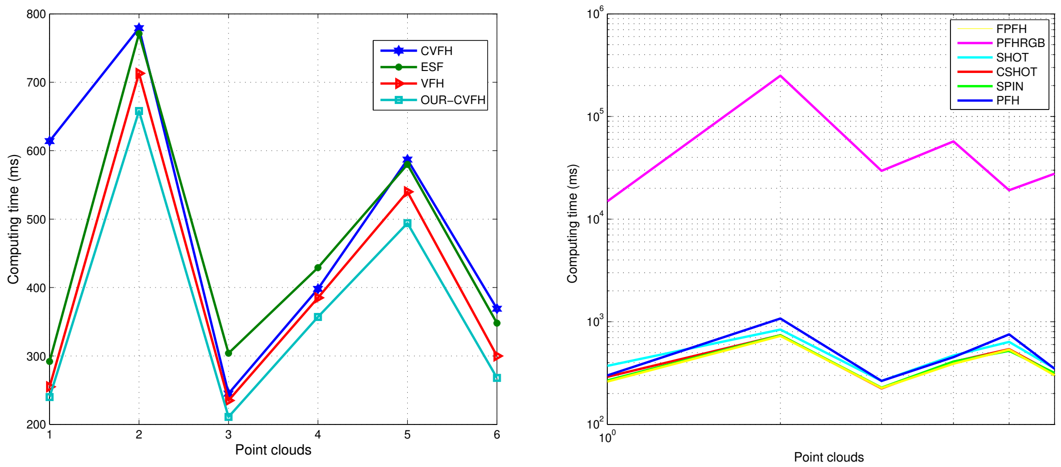

(a) training data (b) testing data

Figure 3.8: The sample objects from Washington RGBD dataset used in the experiments.

Time complexity

As shown in Figure 3.9b, the PFHRGB descriptor requires a very large calculation time compared to the other local descriptors. This result is expected because the PFH descriptor is too slow so if we add the RGB processing the computing time will increase. While, in the case of the global descriptors, Figure 3.9a shows that the computing time is substantially similar in all cases.



(a) global descriptors (b) local descriptors

Figure 3.9: Computing time of PCL's descriptors. Local descriptor plot is represented using the logarithmic scale.

Recognition rate

In order to evaluate our 3D dataset, the proposed recognition pipeline, and the most existing PCL's descriptors, we computed the accuracy for PCL's pipeline for both our 3D and Washington RGBD datasets. Moreover, we compared the evaluated results with Alexandre pipeline [5] that disregarded the recognition threshold. Tables 3.2 and 3.3 present the recognition thresholds of our 3D training data and Washington RGBD training data respectively with all local and global descriptors.

	Box	**Tea**	**Flour 1**	**Flour 2**	**Flour 3**	**Cap**
CVFH	6875.09	32271.8	4753.4	92546.8	16016.1	17994.5
ESF	0.00014	0.00158	0.0008	0.00164	0.00063	0.00087
VFH	426.338	1642.24	549.121	1869.5	370.543	631.457
OUR-CVFH	1941.8	38677.7	1615.16	11054.3	2881.06	1886.64
FPFH	69.5121	42.7864	57.345	884.189	36.3748	108.991
PFHRGB	31.3573	136.114	15.9099	132.883	25.5511	21.7996
SHOT	0.00426	0.01189	0.00668	0.03521	0.00479	0.00557
CSHOT	0.02442	0.10227	0.0112	0.04716	0.02543	0.06437
SPIN	0.00148	0.00026	0.00099	0.00490	6.85e-05	0.00125
PFH	26.8462	42.7864	9.14942	8.89827	220.606	5.92566

Table 3.2: Recognition thresholds of our training set using 3D SIFT.

	Cap	**Food-box**	**Food-can**	**Greens**
CVFH	14306.5	6709.36	624.81	3908.84
ESF	0.000279879	0.000393686	0.00222502	0.000378277
VFH	396.138	1088.58	2466.85	937.507
FPFH	64.9115	432.27	610.522	95.2613
PFHRGB	2.70465	53.1744	11.8484	5.20018
SHOT	0.0542548	0.146218	0.0519007	0.00992696
CSHOT	0.00579158	0.0182675	0.0442323	0.00985572
SPIN	0.000164186	0.00106468	0.00332231	0.000265552
PFH	23.2162	212.57	164.335	17.4079

Table 3.3: Recognition thresholds of the Washington RGBD training set using 3D SIFT.

As shown in Figure 3.10, PFHRGB, and CSHOT descriptors perform better than their original versions PFH and SHOT. This result is due to the use of color information. The FPFH is an extension of the PFH that allows lowering the time complexity. However, the PFH remains better than FPFH in terms of recognition rate. Moreover, the results show that the PFHRGB is the best descriptor, since the value of the accuracy is equal to 100%, thus representing a perfect classification. But, it requires a significant calculation time. In fact, we can use the CSHOT descriptor that represents a compromise between recognition rate and time complexity. Figure 3.10 shows also, that spin image and SHOT are good descriptors since they present the accuracy value equal to 91%. They can be used in the case where the point clouds do not contain the RGB information. Moreover, our pipeline which employs the recognition threshold can classify the object classes more precisely than Alexandre pipeline that was limited only to the Euclidean distance. Another interesting result is that our 3D dataset shows good results, due to the 3D object acquisition. Unlike Washington RGBD dataset that saves only one Kinect frame of the object. In fact, we can probably deduce that some descriptors which are based on the whole cluster or object geometry such as OUR-CVFH gives good results on our 3D dataset because it represents a 360° view of the object. Contrary, VFH descriptor provides good results instead of CVFH and OUR-CVFH extensions when using Washington dataset. However, Washington RGBD data remains the best in term of its simplicity of features processing, just it must be used with many large numbers of views for each object class.

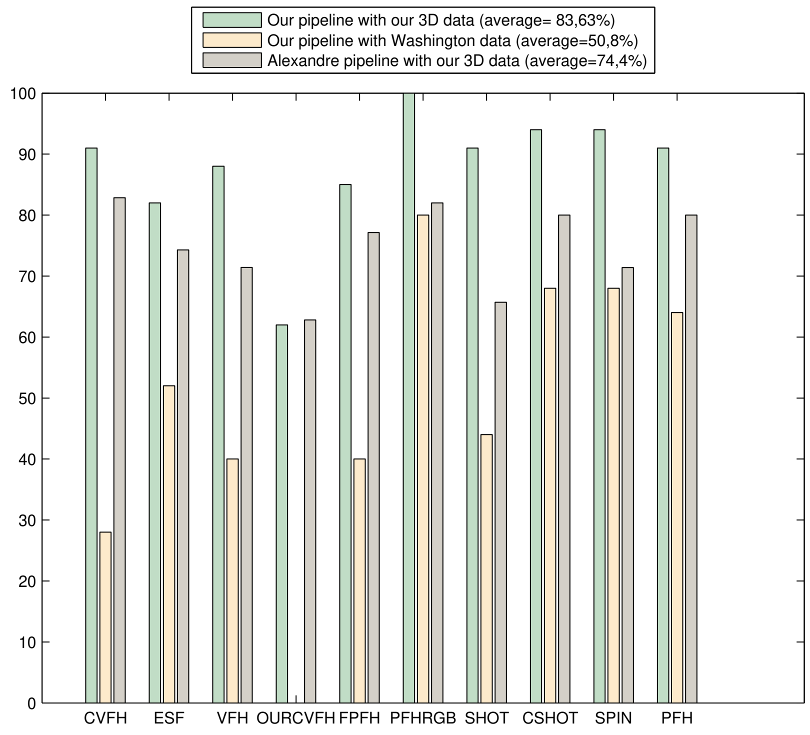


Figure 3.10: Accuracy of PCL's descriptors.

3.3 Conclusion

In this chapter, we reviewed the state-of-the-art of 2D and 3D object recognition approaches, then we proposed a new 3D recognition pipeline based on PCL's descriptors as well as the recognition threshold to perform the object recognition task. First, we provided a new dataset of 3D real-world objects which are captured from multiple Kinect frames using both RGB-Demo software as well as mobile hardware. Then, we evaluated the 3D descriptors implemented in the PCL library by proposing a new 3D recognition pipeline. This later employed a recognition threshold for rejecting misclassified objects. The experimental results show that our proposed pipeline is able to produce good results. Also, we mentioned that our dataset that contains 3D shaped objects of different view acquisition is better than the objects acquired from a single Kinect frame. The main conclusions are:

- PFHRGB requires a very large calculation time but it represents a good descriptor in term of recognition rate;
- PFHRGB and CSHOT are the best descriptors in term of the recognition rate. This is due to their use of color information;
- FPFH descriptor is the amelioration of PFH descriptor in term of computational time but PFH remains better than FPFH in term of recognition rate;

- Spin Image and SHOT are also good descriptors, in the case where the point clouds do not contain the RGB information;
- VFH descriptor provides good results instead of CVFH and OUR-CVFH extensions when using Washington dataset;
- CSHOT represents a compromise between recognition rate and time complexity.

In future work, we will expand our 3D real object data in order to put it available for 3D applications. Then, we will propose new approaches based on the best descriptors and deep learning methods.

Chapter 4

Contributions to 2D/3D Object Recognition and Categorization

"The spaces between the perceiver and the thing perceived can [...] be closed with a shout of recognition".

Timothy Findley

Contents

OBJECT classification is arguably the most important task in the field of computer vision. It provides to machines the ability to see and understand the objects in their surroundings. There exists a huge variety of object classification approaches that can be found in everyday life. Starting with robots in industrial environments, robots in the medical care sector, and up to autonomous systems such as home-assistant robots. An object classification system uses training datasets containing known and labeled objects then extracts different types of information depending on the classification task. This information can describe colors, edges, geometric forms, and Bag of Words. After that, the system performs the classification using machine learning or more recently deep learning techniques. In general, for any new seen object the same information are gathered and compared to the training datasets to find the most suitable classification. In this chapter, we propose several local and global approaches for classifying both 2D/3D objects using various information [72, 179, 291, 294]. For that, we describe a 2D object database and 3D point clouds with 2D/3D local descriptors which we quantify with the k-means clustering algorithm for obtaining the BoWs. Moreover, we develop a new global descriptor called VFH-Color that combines the original version of Viewpoint Feature Histogram (VFH) descriptor [201] with the color quantization histogram, thus adding the appearance information that improves the recognition rate. Then, we train separately these descriptors with Deep Belief Network. In summary, our main contributions are:

1. we describe an object database with SURF features which are quantified with the k-means clustering algorithm to make the 2D BoWs;

2. we describe a point cloud with Spin Image features which we quantify with the k-means clustering algorithm to generate the 3D BoWs;

3. we propose VFH-Color descriptor that combines both the color information and geometric features extracted from the previous version of VFH descriptor. We extract the color information for point cloud data, and then we use the color quantization technique to obtain the color histogram which is combined with VFH histogram.

4.1 Introduction

A human can search and find an object visually in a cluttered scene. It is a very simple task for the human to pick up an object and place it in the required place while avoiding obstacles along the path, and without damaging the fragile objects. These simple and trivial tasks for humans become challenging and complex for robots and can overcome their capabilities. The majority of pick-up and drop applications through robots are performed in fully known and structured environments. The key question that arises in this context is how robots can perform as well as humans in these tasks when the structure of the environment is varied? Human vision is extremely robust and can easily classify objects among tens of thousands of possibilities [24] within a fraction of a second [183]. The human system is able to tolerate the tremendous changes in scale, illumination, noise, and viewing angles for object recognition. Contrary to the human vision, the object recognition is a very complex problem and still beyond the capabilities of artificial vision systems. This contrast between vision systems and the human brain for performing visual recognition and classification tasks gave rise to the development of several approaches to visual recognition. The ability to recognize and manipulate a large variety of objects is critical for mobile robots. Indoor environment often contains several objects on which the robot should make different actions such as "pick-up the remote control" and "drop it inside the box!" So, how to represent and classify objects to be recognized by robots?

In this chapter, we suggest new approaches for 2D/3D object recognition and categorization for mobile robotic applications. We introduce two different recognition pipelines, the first one relies on 2D/3D detectors and descriptors which are quantified with a k-means algorithm to obtain 2D/3D BoWs, while the second one uses our new 3D global descriptor called VFH-Color. Figure 4.1 summarizes the main steps of 2D/3D BoWs approaches.

1. **Training set:** represents a set of data (i.e., images or point clouds) used in our experiments. Training means, creating a dataset with all the objects which we want to recognize.

2. **Keypoint extraction:** is the first step of our approach where keypoints are extracted from input data. It reduces the computational complexity by identifying particularly those regions of images which are important for descriptors, in term of high information density.

3. **Keypoint description:** once keypoints are extracted, descriptors are computed on the obtained keypoints and these form a description which is used to represent the data.

4. **Vocabulary:** after the extraction of descriptors, the approach uses the vector quantization technique to cluster descriptors in their feature space. Each cluster is considered as "visual word vocabulary" that represents the specific local pattern shared by the keypoints in this cluster.

5. **Bag of Words:** is a vector containing the (i.e., weighted) count or occurrence of each visual word in the data which is used as the feature vector in the recognition and classification tasks.

6. **Classificiation:** all data in the training set are represented by their BoWs vectors which represent the input of DBN classifier.

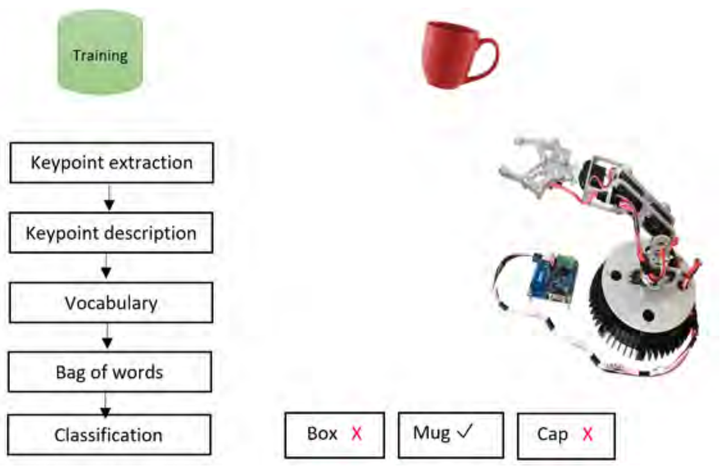


Figure 4.1: Overview of 2D/3D BoWs approaches.

For the global pipeline, we present a new VFH-Color descriptor that combines both the color information and the geometric features extracted from the previous version of VFH descriptor. Figure 4.2 summarizes the main steps of the global approach.

1. **Training set:** represents a set of point clouds used in our experiment.

2. **3D point description:** extracts the color information for point cloud data, then uses the color quantization technique to obtain the color histogram which is combined with VFH histogram extracted from the previous version of VFH descriptor.

3. **Classificiation:** all point clouds in training set are represented by their VFH-Color features and are provided as the input to DBN classifier.

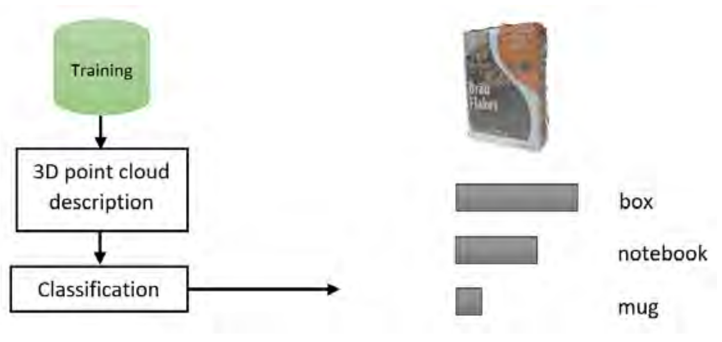


Figure 4.2: Overview of 3D global approach.

4.1.1 Object representation

2D Bag of Words (2D BoWs)

Recently, appearance-based methods have been successfully applied to the problem of object recognition. These methods typically proceed with two phases. In the first phase, a model is constructed from a set of training images that includes the appearance of the object under different illuminants, scales, and multiple instances. Whereas, in the second phase, the methods try to extract parts from the input image through segmentation or by using the sliding windows over the whole image. The methods then compare extracted parts of the input image with the training set. A popular strategy of appearance-based methods is the BoWs. BoWs is inspired by text-retrieval systems that count how many times a word appears in a document. It aims to represent an image as an orderless set of local regions. In general, local regions are discretized into a visual vocabulary. This method obtains excellent results in image classification [11], image retrieval [281], object detection as well as scene classification [85].

2D Speeded-Up Robust Features (SURF) detector Keypoints are important features that are becoming more and more widespread in image analysis. The Speeded-Up Robust Features (SURF) [14, 15] is based on the same steps and principles of SIFT detector [141], but it utilizes a different scheme and provides better results than those obtained with SIFT extractor. Before applying the detector of SURF, we divide the image into small sub-images with the integral images. Given an image $\mathrm{I}(x, y)$, the integral image at any location (x, y) in I is the sum of all the pixels to left of it and above it, including itself. This can be stated mathematically as:

$$\mathrm{S}(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} \mathrm{I}(i, j) \tag{4.1}$$

Then the goal is to compute a set of feature points with their characteristic scales and orientations. The points of the SURF detector are computed with the determinant of the Hessian matrix measuring the local changes around the point. We have to maximize this determinant so that the pixels where it is computed are salient points. Likewise, this determinant is used to determine the scale σ. The Hessian is given with:

$$H(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix} \tag{4.2}$$

Where p(x,y) is an image and:

$$\begin{aligned} L_{xx} &= S * G_{xx}(\sigma) \\ L_{xy} &= S * G_{xy}(\sigma) \\ L_{yy} &= S * G_{yy}(\sigma) \end{aligned} \tag{4.3}$$

Furthermore, the Gaussian G is approximated with a box filter of size 9×9 at the scale $\sigma = 1.2$. Because of the importance of the invariance to the rotation, the orientation of each feature point needs to be detected. Around each feature point, we compute the Haar wavelet responses in the x and y directions in a circle of size 6σ where σ is the scale. Next, we convolve this circle with a Gaussian kernel giving us a matrix of values in the horizontal and vertical axes. The orientation is calculated by summing all values of this matrix with a sliding orientation window of size $\pi/4$. While horizontal and vertical responses within the window are summed and the two summed responses then give a local orientation vector.

2D Speeded-Up Robust Features (2D SURF) descriptor SURF descriptor provides a unique and robust description of a feature that can be generated on the area surrounding a keypoint. SURF descriptor is based on Haar wavelet responses and can be calculated efficiently with integral images. SURF describes an interesting area with size 20, then each interest area is divided into 4×4 sub-areas and is described by the values of a wavelet response in the x and y directions. The interest areas are weighted with a Gaussian centered at the keypoint for being robust in deformations and translations. For each sub-area, a vector v is calculated, based on 5×5 samples. The descriptor for keypoint consists of 16 vectors for the sub-areas being concatenated. Finally, the descriptor is normalized, to achieve invariance to contrast variations that will represent themselves as a linear scaling of the descriptor.

Visual vocabulary Once the keypoint descriptors are obtained, the approach imposes a quantization on the feature space of these descriptors. The standard pipeline to obtain "visual vocabulary" is also called "codebook" which consists of (i) collecting a large sample of a local feature, and (ii) quantizing the feature space according to their statistics. Most vector quantization or clustering algorithms are based on hierarchical or iterative square error partitioning methods. Hierarchical methods organize data on groups which can be displayed in the form of the tree. Whereas, square-error partitioning algorithms attempt to obtain which maximizes the between-cluster scatter or minimizes the within-cluster scatter. In our work, we use a simple k-means clustering algorithm. The "visual words" or "codevector" represent the k cluster centers. A vector quantizer takes a feature vector and maps it to the index of the nearest codevector in the codebook using the Euclidean distance.

Bag of Words BoWs is generated by computing the count or occurrence of each visual word in the image which is used as the feature vector in the recognition and classification tasks. As shown in Figure 4.3, the black dot represents 240 SURF keypoints of the object bottle. Next, the approach computes the SURF descriptor on each keypoint and fixes the number of visual words (W1, W2, W3, and W4) denoted as cluster centers. The sampled features are clustered in order to optimize the space into a discrete number of visual words. Then, a bag of visual words histogram can be used to summarize the entire image. It counts the occurrence of each visual word in the image.

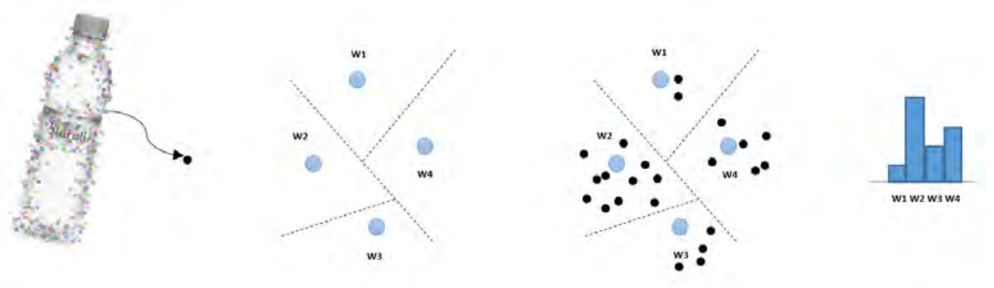


Figure 4.3: The schematic illustrates visual vocabulary construction and word assignment.

3D Bag of Words (3D BoWs)

3D Scale-Invariant Feature Transform (3D SIFT) SIFT is deployed in the field of computer vision to detect and describe regions in an image and identify similar elements between varying images. This process is called "matching". It was originally developed for 2D images [141] and was adapted by the community of the PCL library to 3D point clouds by replacing the role of the intensity of a pixel in the original algorithm by the principal curvature of a point within the 3D cloud. 3D SIFT detector [212] uses the Difference of Gaussian (DoG) function to extract the extrema points in both spatial and scale dimensions. Section 3.2.8 provides more details about 3D SIFT detector.

Spin Image (SI) descriptor The spin image [91] was proposed to describe 3D keypoints. It represents the oldest PCL descriptor that has been around since 1997, but it still used in some object recognition applications. It was originally designed to describe surfaces made by vertices, edges, and polygons, but it has been since adapted for point clouds. SI used a cylindrical support structure, centered at the point, with a given radius and height, and aligned with the normal. This support structure is divided radially and vertically into volumes. For each one, the number of neighbors lying inside is added up, eventually producing a descriptor. Then, the descriptor performs weighting and interpolation steps to improve the result. The final descriptor can be represented as a grayscale image where dark areas correspond to volumes with higher point density.

Visual vocabulary After describing each of the point clouds inside a class with the SI descriptor, we need to make the visual categorization using the probabilistic approach. The method we use consists of applying a quantization operation with the k-means clustering and constructs visual words with the well-known method of the bag of features.

Bag of Words Instead of considering each feature point a visual word, we consider thanks to the quantization that each of the clusters' center represents a word. BoWs algorithm consists of computing the number of occurrences of each word in the model database. It is like a probability of the number of words inside the class of objects.

Viewpoint Feature Histogram-Color (VFH-Color)

The development of 3D perception sensors like Microsoft Kinect has triggered a wide availability and use of 3D point clouds. It creates suitable data representations that facilitate object detection, recognition, and categorization. In 3D local descriptors, each point is described by its local geometry. They are developed for specific applications such as object recognition, and local surface categorization. On the other hand, the 3D global descriptors describe object geometry. They are not computed for individual points, but for a whole cluster instead. The global descriptors are high-dimensional representations of object geometry. They are more efficient in object recognition, geometric categorization, and shape retrieval. In general, 3D object recognition methods that use the Kinect camera attain different perspectives. Approaches based on local descriptors are used thanks to the color information, unlike approaches based on global appearance which are more practical thanks to the use of depth information. Here, we present a new VFH-Color descriptor that combines both the color information and the geometric features extracted from the previous version of VFH descriptor (Figure 4.4).

VFH [201] computes a global descriptor of the point cloud and consists of two components: the component associated with FPFH and a viewpoint direction component. VFH aims to combine the viewpoint direction directly into the relative normal angle calculation in the FPFH descriptor [199]. The viewpoint-dependent component of the descriptor is a histogram of the angles between the vector $(p_c - p_v)$ and each point's normal. This component is binned into a 128-bin histogram. The other component is a simplified point feature histogram (SPFH) estimated for the centroid of the point cloud, and an additional histogram of distances of all points in the cloud to the cloud's centroid. The three angles (α, ϕ, θ) with the distance d between each point and the centroid are binned into a 45-bin histogram. The total length of the VFH descriptor is the combination of these two histograms and is equal to 308 bins.

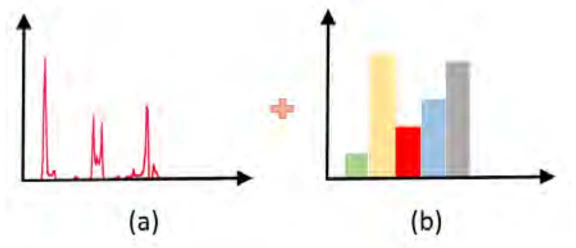


Figure 4.4: VFH-Color. (a) VFH descriptor. (b) Color quantization.

Color quantization is a vector quantization that aims to select K vectors in N dimensional space in order to represent N vectors from that space (K << N). In general, color quantization is applied to reduce the number of colors in a given image while maintaining the visual appearance of the original image. Figure 4.5 shows the color quantization which is applied in a 3-dimensional space RGB and follows these steps:

1. extract RGB features for each point from the point cloud data;
2. obtain the matrix of RGB features (i.e., number of points$\times$3);
3. compute k-means algorithm for the RGB matrix in order to generate the codebook (cluster centers);
4. finally, count the occurrence of each codebook in the point cloud.

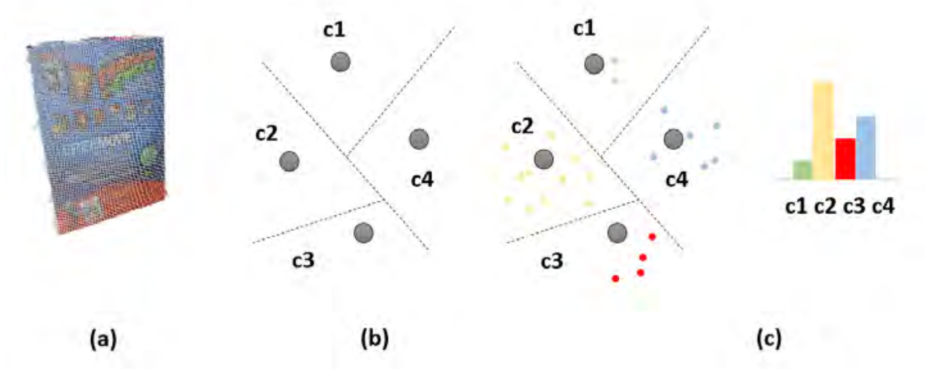


Figure 4.5: Color quantization process. (a) Point cloud data. (b) Codebook (C1, C2, C3, and C4) denote cluster centers. (c) RGB features are clustered in order to optimize the space. The histogram counts the occurrence of each codebook in the point cloud.

The codebook size represents the bins of color quantization histogram. According to the experiments which are shown in Figure 4.6, we fix the codebook size to 100 bins providing the highest value of accuracy. Therefore, VFH-Color histogram concatenates 308 values of original VFH descriptor and 100 values of color quantization histogram, thus giving the total size of 408 values.

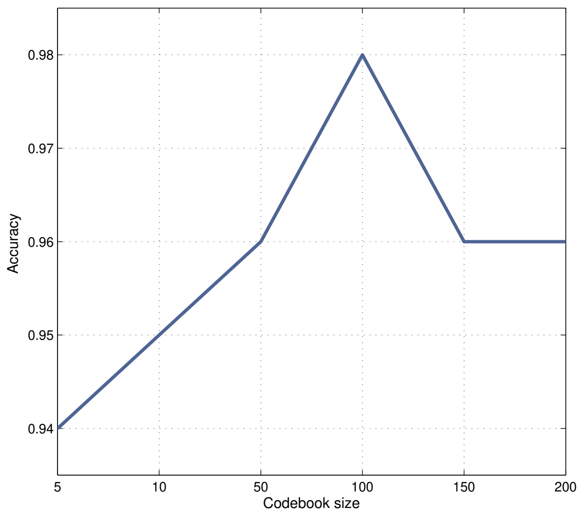


Figure 4.6: The classification performance with respect to the codebook size.

4.1.2 Object classification

Deep Belief Network

Restricted Boltzmann Machine (RBM) is a two-layer network in which stochastic visible units are connected to stochastic hidden units using symmetrically weighted connections. In the typical RBM, both visible and hidden units are binary. Since our features are real-valued data, we use the GBRBM type in which the visible units are continuous. By stacking several RBMs, we form a multilayer stochastic generative model called Deep Belief Network (DBN). DBN consists of one visible layer and many hidden layers. Each hidden layer unit learns a statistical relationship between the units in the lower layer. By this way, DBN can be efficiently trained using greedy layer-wise training algorithm. After, generative pre-training, supervised backpropagation can be used to fine-tune the features classification. We refer to this architecture as BP-DBN or GDBN (the details of GDBN are provided in Section 2.2.3).

Support Vector Machines

Support Vector Machines (SVMs) [31] are a useful method for data classification. They are based on the concept of decision planes which separate between a set of objects that have different class memberships.

Let's consider a training set of instance-label pairs (x_i, y_i) where $i = 1, ..., l$, $x_i \in \mathrm{R}^n$, and $y \in (-1, 1)^l$, the optimization problem is defined as:

$$min_{w,b,\xi}(\frac{1}{2}\mathrm{W}^{\mathrm{T}}\mathrm{W} + \mathrm{C}\sum_{i=1}^{l}\xi_i) \tag{4.4}$$

subject to

$$y_i(\mathrm{W}^{\mathrm{T}}\phi(x_i) + b) \geq 1 - \xi_i \tag{4.5}$$

with $\xi_i \geq 0$ and $\mathrm{C} > 0$.

x_i are mapped into a higher dimensional space by the use of function ξ. SVMs try to find a linear separating hyperplane with the maximal margin in this higher dimensional space. C represents the penalty parameter of the error term.

In our work, we are interested in multi-class classification. For this purpose, we use C-Support Vector Classification (C-SVC) for two-class and multi-class classification. $\phi(x_i)$ maps x_i into a higher dimensional space. Because of the possible high dimensionality of the vector variable W, generally the following dual problem is solved:

$$min_{\alpha} \qquad \frac{1}{2}\alpha^{\mathrm{T}}\mathrm{Q}\alpha - e^{\mathrm{T}}\alpha \tag{4.6}$$

subject to

$$y^{\mathrm{T}}\alpha = 0 \tag{4.7}$$

with $0 \leq \alpha_i \leq \mathrm{C}$ and $i = 1, ..., l$.

With $e = [1....1]^{\mathrm{T}}$ represents the vector of all ones, $\mathrm{Q}_{ij} \equiv y_i y_j \mathrm{K}(x_i, x_j)$ as well as $\mathrm{K}(x_i, x_j) \equiv \phi(x_i)^{\mathrm{T}}\phi(x_j)$ denotes the kernel function, and Q is an l by l positive semi-definite matrix. In this case, training vectors x_i are mapped into a higher (infinite) dimensional space.

After solving Equation 4.4, and using the primal-dual relationship, the optimal W should satisfy the following equation.

$$\mathrm{W} = \sum_{i=1}^{l} y_i\alpha_i\phi(x_i) \tag{4.8}$$

The decision function is defined as:

$$sgn = (\mathrm{W}^{\mathrm{T}}\phi(x) + b) = sgn(\sum_{i=1}^{l} y_i\alpha_i\mathrm{K}(x_i, x) + b) \tag{4.9}$$

These parameters, support vectors, and other information such as kernel are stored in the model for prediction.

4.2 Experimental results

4.2.1 Datasets

ALOI dataset

Amsterdam Library of Object Images (ALOI) [63] dataset is an image collection of 1000 small objects recorded for recognition task. 111,250 images are captured by Sony DXC390P 3CCD cameras varying viewing angle, illumination angle and illumination color for each object, and additionally images are captured wide-baseline stereo images.

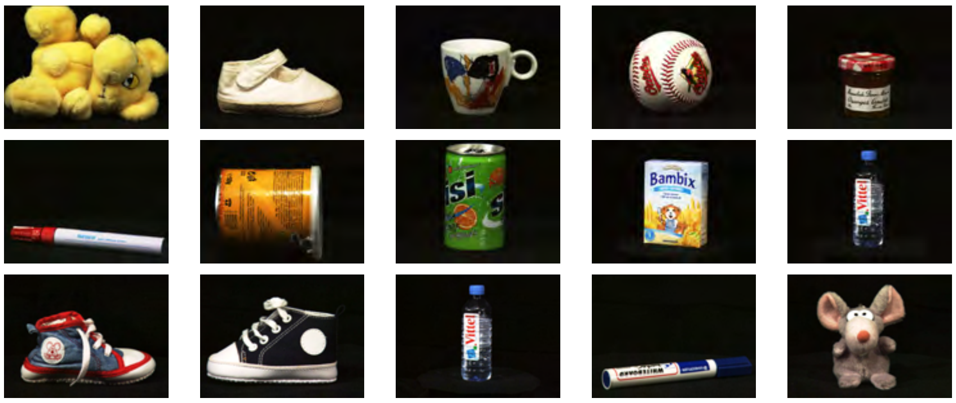


Figure 4.7: The sample images extracted from Amsterdam Library of Object Images (ALOI) dataset.

Washington RGBD dataset

Washington RGBD dataset [111] is a large dataset built for 3D object recognition and categorization applications. Section 3.2.3 provides technical information about the dataset. In this experimentation, we selected 600 point clouds from 10 classes: a ball, a bowl, a cap, a cereal box, a mug, a food box, a food can, a notebook, a plate, and a shampoo (see Figure 4.8).

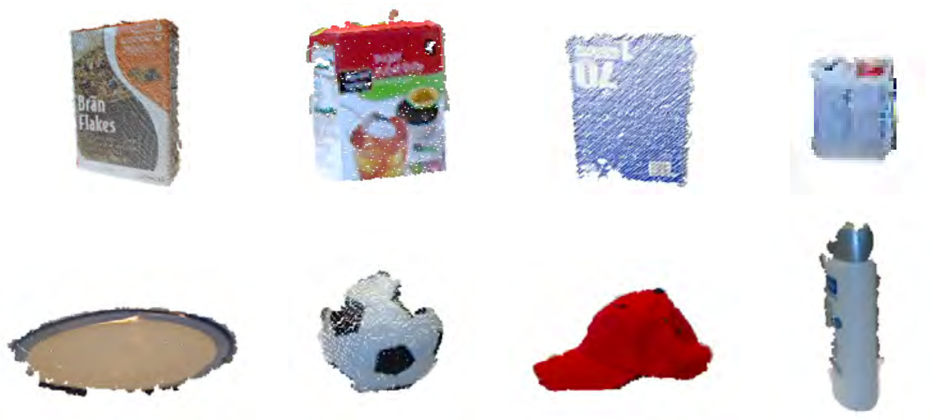


Figure 4.8: The sample point clouds extracted from Washington RGBD Dataset.

4.2.2 2D/3D object classification results

Experimental setup

DBN aims to allow each RBM model in the sequence to receive a different representation of the data. In other words, after RBM has been learned, the activity values of its hidden units are used as the training data for learning a higher-level RBM. The input layer has a number N of units, equal to the size of sample data x (i.e., size of 2D/3D features). The number of units for hidden layers, currently, are pre-defined according to the experiment. We fixed DBN with three hidden layers $h1$, $h2$ and $h3$. The general DBN characteristics are provided in Table 4.1.

Characteristics	**Values**
Hidden layers	3
Hidden layer units	600
Learn rates	0.01
Learn rate decays	0.9
Epochs	200
Input layer units	size of descriptor

Table 4.1: DBN characteristics that are used in our experiments.

Comparing shallow architectures with deep learning methods, we train the SVM classifier (C=1.0, kernel='rbf') for multi-class classification. We chose the RBF's kernel, a nonlinear one, mapping the features to the labels even when the relationship between them is nonlinear.

Evaluation metrics

A binary/multi-class classifier predicts all data of testing as either positive or negative. This classification (or prediction) produces four outcomes: true positive, true negative, false positive, and false negative.

- True positive (TP): it represents the true object that was classified as positive;
- False negative (FN): it represents the true object that was classified as negative;
- True negative (TN): it represents the false object that was classified as negative;
- False positive (FP): it represents the false object that was classified as positive.

Out of these four outcomes, we compute the recall, the precision, the f1-score, the accuracy (ACC), and the confusion matrix.

The recall (also sensitivity or true positive rate) of classifier defines the occurrence of correct positive results among all positive samples available during the test. It is estimated as:

$$recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4.10}$$

The precision (also positive predicted value) of classifier defines the fraction of retrieved instances that are relevant. It is estimated as:

$$precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4.11}$$

The f1-score (also called f-measure) is the weighted average of recall and precision. Therefore, it takes both false positives and false negatives into account. It is estimated as:

$$f1-score = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.12}$$

The accuracy (ACC) of classifier defines the number of samples correctly classified (true positives plus true negatives) and is evaluated by the formula:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{4.13}$$

The confusion matrix contains information about actual and predicted classifications done by a classification system. It is a table formed by counting the number of the four outcomes of a classifier. The following table shows the confusion matrix for a two-class classifier.

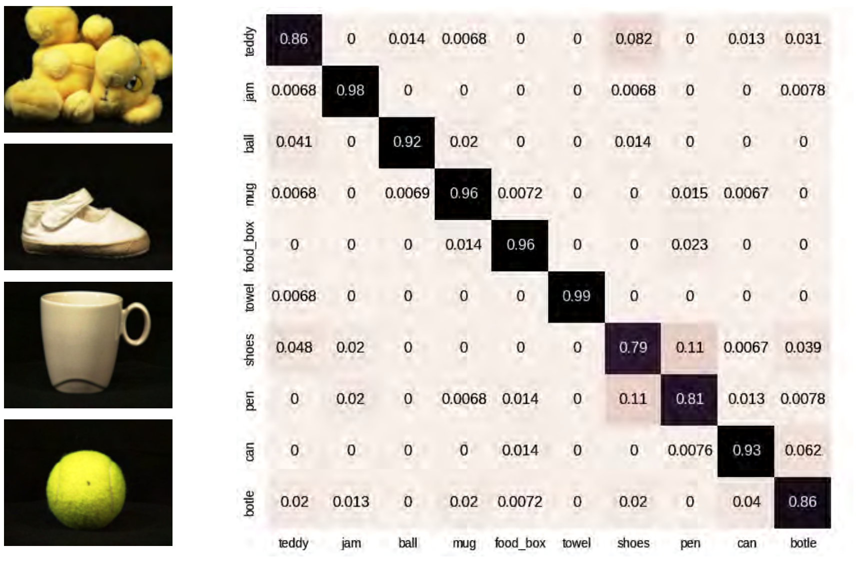
	Predicted value	
Actual value	True Positive	False Negative
	False Positive	True Negative
Total	P	N

Table 4.2: Confusion matrix.

2D Bag of Words results

Images contain local points or keypoints defined as salient region patches which represent rich local information of the image. We used SURF to automatically detect and describe keypoints from images. Then, we used the vector quantization method in order to cluster the keypoint descriptors in their feature space into a large number of clusters using the k-means clustering algorithm. In Figure 4.10a, we test in a set of experiments the impact of the number of clusters on classifier accuracy and we select k=1500 as the size of the codebook (i.e., a number of visual words) that represents the best accuracy value. We conduct the experiments on ALOI dataset on which we select ten categories: a teddy, a jam, a ball,a mug, a food box, a towel, shoes, a pen, a can, and a bottle. Figure 4.7 shows some examples from ALOI dataset which are used in our experiments.

As shown in the confusion matrix (Figure 4.9b) and Table 4.3, the classes which are consistently misclassified are a teddy, a ball, shoes, a can, a mug, and a bottle which are very similar in appearance (Figure 4.9a). The results show also that 2D BoWs approach which is based on SURF features works perfectly with the accuracy rate of 91%. BoWs representation encodes only the occurrence of the appearance of the local patches and ignores the object geometry. The lack of geometric features can provide some misclassification especially when the objects are similar in appearance. In Table 4.8, we report accuracy values for 2D BoWs with both SVM and DBN classifiers. The first row reports the accuracy value of SVM whereas the second row shows the accuracy value of DBN. We notice that the combination of 2D Bag of Words and DBN outperforms the 2D BoWs with SVM and rises steadily from 88.86% to 90.83%. This result shows the power of deep learning architectures that learn multiple levels of representation depending on the depth of the architecture.



(a) Objects (b) Confusion matrix

Figure 4.9: Confusion matrix and the objects which are misclassified using 2D Bag of Words.

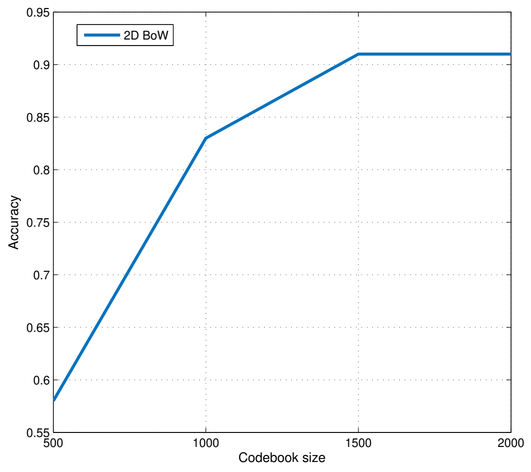
Classes	**Metrics**			
	confused classes	**f1-score**	**recall**	**precision**
teddy	ball, mug, shoes, can, and botle	86%	86%	87%
jam	teddy, shoes, and botle	96%	98%	95%
ball	teddy, mug, and shoes	95%	92%	98%
mug	teddy, ball, food box, pen, and can	95%	96%	93%
food box	mug, and pen	96%	96%	96%
towel	teddy	100%	99%	100%
shoes	teddy, jam, pen, can, and botle	78%	79%	77%
pen	jam, mug, food box, shoes, can, and botle	82%	81%	84%
can	food box, pen, and botle	92%	93%	92%
botle	teddy, jam, mug, food box, shoes, and can	86%	86%	85%
Average	–	**91%**	**91%**	**91%**

Table 4.3: The performance of 2D Bag of Words.

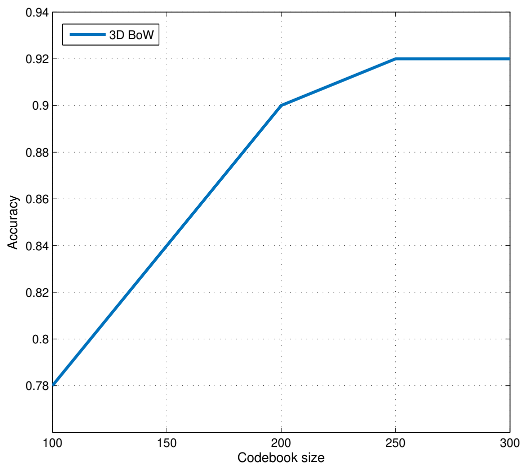
3D Bag of Words results

After extracting the Spin Image for the set of point clouds, we constructed a shape dictionary whose size is fixed at k=250 (Figure 4.10b), by clustering all spin image acquired from the whole training set with the k-means method. For each bin, a representative local 3D feature description is required. These descriptions are taken from the centroids of each cluster (i.e., visual words) determined by k-means clustering on precomputed SI descriptors.

Figure 4.11 represents the confusion matrix across all 10 classes. Most model's results are reasonable showing that 3D BoWs can provide high-quality features. The classes that are



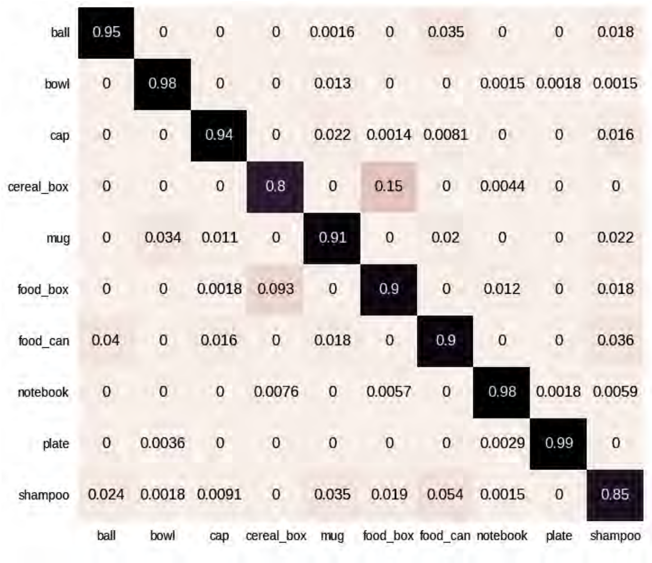
(a) 2D BoWs



(b) 3D BoWs

Figure 4.10: The classification performance with respect to the codebook size for both 2D BoWs and 3D BoWs.

consistently confused are a ball with a mug and a can, a bowl with a mug, a notebook and a plate, a food box with a cereal box and a notebook, which are very similar in shape. Table 4.4 illustrates the performance metrics of 3D BoWs that encodes only the surface shape of 3D point clouds thanks to the use of SI descriptors.



	ball	bowl	cap	cereal_box	mug	food_box	food_can	notebook	plate	shampoo
ball	0.95	0	0	0	0.0016	0	0.035	0	0	0.018
bowl	0	0.98	0	0	0.013	0	0	0.0015	0.0018	0.0015
cap	0	0	0.94	0	0.022	0.0014	0.0081	0	0	0.016
cereal_box	0	0	0	0.8	0	0.15	0	0.0044	0	0
mug	0	0.034	0.011	0	0.91	0	0.02	0	0	0.022
food_box	0	0	0.0018	0.093	0	0.9	0	0.012	0	0.018
food_can	0.04	0	0.016	0	0.018	0	0.9	0	0	0.036
notebook	0	0	0	0.0076	0	0.0057	0	0.98	0.0018	0.0059
plate	0	0.0036	0	0	0	0	0	0.0029	0.99	0
shampoo	0.024	0.0018	0.0091	0	0.035	0.019	0.054	0.0015	0	0.85

Figure 4.11: Confusion matrix of 3D Bag of Words.

Global pipeline results

VFH-Color descriptor combines both the color information and the geometric features extracted from the previous version of VFH descriptor. We extract the color information for point cloud data, then we use the color quantization technique to obtain the color histogram which is combined with VFH histogram. For each point cloud, we extract two types of features: 1) geometric features extracted from Viewpoint Feature Histogram (VFH) (308 bins), and 2) color features extracted from color quantization (100 bins). These features are then combined into a single vector, being 308+100=408 dimensional.

Classes	**Metrics**			
	confused classes	**f1-score**	**recall**	**precision**
ball	mug, food can, and shampoo	94%	95%	94%
bowl	mug, notebook, plate, and shampoo	97%	98%	96%
cap	mug, food box,food can, and shampoo	95%	94%	96%
cereal box	food box, and notebook	84%	80%	89%
mug	bowl, cap, food can, and shampoo	91%	91%	91%
food box	cap ,cereal box,notebook, and shampoo	87%	90%	84%
food can	ball, cap,mug, and shampoo	89%	90%	88%
notebook	cereal box, food box, plate, and shampoo	98%	98%	98%
plate	bowl, and notebook	99%	99%	100%
shampoo	ball, bowl, cap, mug, food box, food can , and notebook	87%	85%	88%
Average	–	**92%**	**92%**	**92%**

Table 4.4: The performance of 3D Bag of Words.

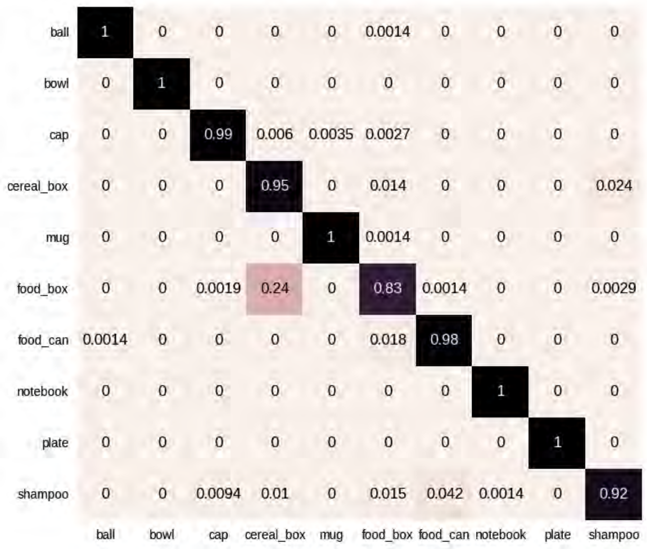


Figure 4.12: Confusion matrix of global pipeline using VFH descriptor.

Before evaluating VFH-Color descriptor, we test the global recognition pipeline with the basic version of VFH descriptor. According to Figure 4.12, VFH features confuse between several object point clouds such as a ball with a food box, a cap with a cereal box, a mug, and a food box, also a cereal box with a shampoo, a food box with a food can and a shampoo, as well as a shampoo with a cap, and a notebook. Tables 4.5 and 4.6 compare the evaluation metrics between VFH descriptor and VFH-Color descriptor respectively. It's shown that VFH-Color descriptor represents the highest precision value 99% compared to the VFH descriptor one that is equal to 97%. This result confirms that our VFH-Color descriptor can classify 3D objects with the low false positive rate. Figure 4.13 represents the confusion matrix across all 10 classes. Most model's results are very reasonable showing that VFH-Color can provide meaningful features. The classes that are consistently misclassified are a mug with a cap,a cereal box, a food box, and a shampoo, also a cap, a mug, and a food can which are very similar in appearance and shape.

Moreover, we evaluate the performance of VFH-Color against the previous version of VFH and CSHOT (SHOTCOLOR). The accuracy using VFH-Color performs 3% better than

Classes	**Metrics**			
	confused classes	**f1-score**	**recall**	**precision**
ball	food box	100%	100%	100%
bowl	–	100%	100%	100%
cap	cereal box, mug, and food box	99%	99%	99%
cereal box	food box, and shampoo	86%	95%	79%
mug	food box	100%	100%	100%
food box	cap, cereal box, food can, and shampoo	88%	83%	94%
food can	ball, and food box	97%	98%	96%
notebook	–	100%	100%	100%
plate	–	100%	100%	100%
shampoo	cap, cereal box, food box, food can, and notebook	95%	92%	97%
Average	–	**95%**	**96%**	**97%**

Table 4.5: The performance of global pipeline using VFH descriptor.

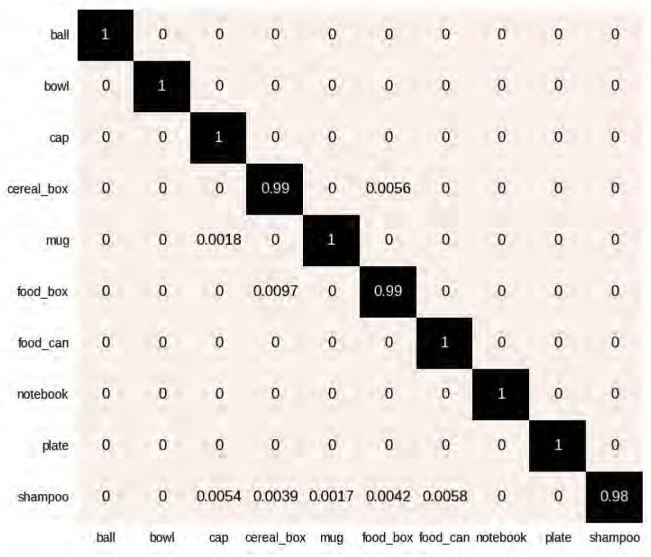


Figure 4.13: Confusion matrix of global pipeline using our VFH-Color descriptor.

Classes	**Metrics**			
	confused classes	**f1-score**	**recall**	**precision**
ball	–	100%	100%	100%
bowl	–	100%	100%	100%
cap	–	100%	100%	99%
cereal box	food box	99%	99%	99%
mug	cap	100%	100%	100%
food box	cereal box	99%	99%	99%
food can	–	100%	100%	99%
notebook	–	100%	100%	100%
plate	–	100%	100%	100%
shampoo	cap, cereal box, mug, food box, and food can	99%	98%	100%
Average	–	**99%**	**99%**	**99%**

Table 4.6: The performance of global pipeline using VFH-Color descriptor.

VFH that models only the geometric features. This result shows the effectiveness of the approach after adding the color information. We also notice that CSHOT (SHOTCOLOR) presents a good accuracy (Table 4.7 and Figure4.14), although this descriptor encounters a problem when it is not able to compute the local reference frame for some point clouds. In this set of experiments, 15% of point clouds from the dataset are not computed with CSHOT (SHOTCOLOR). This problem becomes significant when 3D object recognition is in real-time. Indeed, VFH-Color descriptor can be used in the real-time applications thanks to its estimation for every point cloud as well as its good recognition rate. Table 4.8 shows also that our global pipeline works perfectly with the accuracy rate of 99.63% with DBN architecture that performs the use of SVM classifier. In general, the use of DBN instead of SVM in our approaches increases the accuracy rate thanks to the performance of deep learning algorithms which outperformed the shallow architectures (SVM).

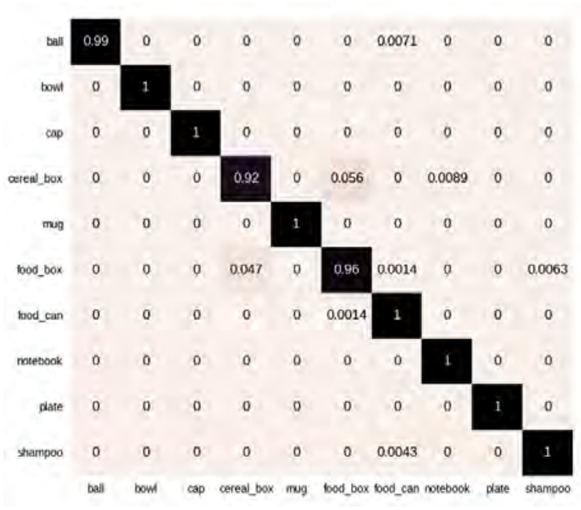


Figure 4.14: Confusion matrix of CSHOT (SHOTCOLOR) descriptor.

Classes	**Metrics**			
	confused classes	**f1-score**	**recall**	**precision**
ball	food can	100%	99%	100%
bowl	–	100 %	100%	100%
cap	–	100%	100%	100%
cereal box	food box, and notebook	93%	92%	95%
mug	–	100%	100%	100%
food box	cereal box,food can, and shampoo	95%	96%	94%
food can	food box	99%	100%	99%
notebook	–	100%	100%	99%
plate	–	100%	100%	100%
shampoo	food can	99%	100%	99%
Average	–	**99 %**	**99%**	**99%**

Table 4.7: The performance of CSHOT (SHOTCOLOR) descriptor.

4.2.3 Comparison to other methods

In this subsection, we compare our contributions to the related state-of-the-art approaches. Table 4.9 shows the main accuracy values and compares our recognition pipelines to the published results [26, 111, 211] and [50, 144]. Lai *et al.* [111] extract a set of features that

Classifier	2D BoWs	3D BoWs	VFH	VFH-Color	CSHOT
SVM	88.86%	86.68%	95.01 %	98.34%	97.21 %
DBN	90.83%	92.03%	96.41 %	**99.63%**	98.63 %

Table 4.8: Accuracy of different proposed approaches using DBN and SVM classifiers.

captures the shape of the object view using a spin image and another set which captures the visual appearance using SIFT descriptors. These features are extracted separately from both depth and RGB images. A recent work by Schwarz *et al.* [211] uses both colorizing depth and RGB images that are processed independently by a CNN. CNN features are then learned using SVM classifier in order to successively determine the category, instance, and pose. The previous approaches [50, 144] used the color-coding depth maps and RGB images for training separately CNN architecture. In our work, we learn our 3D features using DBN with three hidden layers that model a deep network architecture. The results show also that our global pipeline works perfectly with the accuracy rate of 99.63% thanks to the efficiency of our VFH-Color descriptor and outperforms all methods that are mentioned in the state-of-the-art.

Approaches	Accuracy rates
Lai *et al.* [111]	90.6%
Bo *et al.* [26]	84.5%
Eitel *et al.* [50]	91%
Madai *et al.* [144]	94%
Schwarz *et al.* [211]	94.1%
VFH and DBN	96.41%
3D BoW and DBN	92%
VFH-Color and DBN	**99.63%**
CSHOT and DBN	98.63%

Table 4.9: The comparison of 3D object recognition accuracies and PCL descriptors on the Washington RGBD dataset.

4.3 Conclusion

In this chapter, we proposed new approaches for object categorization and recognition in the real-world environment. We used the BoWs that aims to represent images and point clouds as an orderless of local regions that are discretized into a visual vocabulary. Also, we proposed the VFH-Color descriptor which combined geometric features extracted from VFH descriptor and color information extracted from the color quantization method. Then, we learned the 2D and 3D features with DBN. The experimental results on ALOI dataset and Washington RGBD dataset clearly ascertain that the proposed algorithms are able to classify images and 3D point clouds. These results are encouraging, especially that our new VFH-Color descriptor performed the state-of-the-art methods in recognizing 3D objects under different views. Also, our approach improved the recognition rates thanks to the use of color information.

In future work, we will attempt to embed our algorithms in a mobile robot in order for it to recognize and manipulate the real-world objects.

Chapter 5

Generative vs. Discriminative Deep Belief Networks for 3D Object Categorization

"Learning is not attained by chance, it must be sought for with ardor and attended to with diligence".

Abigail Adams

Contents

OBJECT categorization and manipulation are critical tasks for a robot to operate in the household environment. With the availability of the Microsoft Kinect camera, there is a surge of interest in 3D object categorization from point cloud data. Most of the 3D categorization methods use geometric features, BoWs, and shapes extracted from certain projections of the 3D objects. In computer vision, categorization is considered a harder problem than recognition one, since it learns a generic model from instances that belong to the same category. Therefore, categorization approaches should use the geometric features which describe common properties of all category instances. In this chapter, we propose a global approach for representing and learning 3D object categories using global descriptor and deep learning architectures [292, 293]. As global descriptors describe an entire object, a pre-processing step is usually required to remove planes and walls in the 3D scene and then segment it into different objects. After this segmentation step, we extract geometric features from 3D point clouds using the Viewpoint Feature Histogram (VFH) descriptor and then we learn these features with Deep Belief Network (DBN). Thereafter, we evaluate the performance of both generative and discriminative DBN architectures (GDBN/DDBN) for 3D object categorization task using different RBM training techniques which include Contrastive Divergence (CD), Persistent Contrastive Divergence (PCD), and Free Energy in Persistent Contrastive Divergence (FEPCD). GDBN trains a sequence of Restricted Boltzmann Machines (RBMs) while DDBN uses a new deep architecture based on RBMs and the joint density model. The main contributions of this chapter are:

- we propose a new 3D object categorization pipeline based on VFH descriptor and deep learning architectures;
- we segment all the objects which are present in the 3D scene;
- we extract geometric features using VFH descriptor;
- we learn the extracted features with GDBN and DDBN architectures in order to show the difference between generative and discriminative training for 3D object categorization.

5.1 Introduction

Object categorization is the ability to learn a generic model from instance appearance that belongs to the same family or category based upon commonalities and similarities of objects. In contrary to categorization, the challenging task in recognition is to learn a particular model which presents a specific known object instance. In other words, taking the example case of the box, the goal in categorization is to learn the essence of what properties make objects appears as the box. Whereas in recognition, the goal is to know the very specific object instance cereal food box. Therefore, categorization can be seen as the generalization task of classifying unknown objects from unknown viewpoints. On the other hand, recognition is considered as the generalization task of identifying known objects from unknown viewpoints. Consequently, the major goal in the categorization task is the extraction of category-particular features while considering the diversity in instance appearances with each category. A good object categorization approach should be able to handle two main challenges:

1. the extraction of category-specific features which can model all the instances with respect to the intra-category and inter-category variability;

2. learning efficiently the extracted features to perform the classification task.

Our work focuses specifically on 3D object categorization for mobile robotic grasping. As shown in Figure 5.1, we assumed that the acquired data come from a Kinect-2 camera at a viewing distance of roughly 1m and will be saved as point clouds. The objects of interest are placed on a horizontal surface such as a table, a desk, or a kitchen counter. After that, we use a pre-processing step in order to remove the planes and the walls from the scene, then segment it into different objects. For each segmented object, we extract VFH descriptors that encode geometric features, thus describing the common features of all instances in each category. This step is followed by learning the resulting features using effective deep architectures. We evaluate both Generative and Discriminative DBNs (GDBN/DDBN) in the context of object categorization using different RBM training techniques which include Contrastive Divergence (CD), Persistent Contrastive Divergence (PCD), and Free Energy in Persistent Contrastive Divergence (FEPCD).

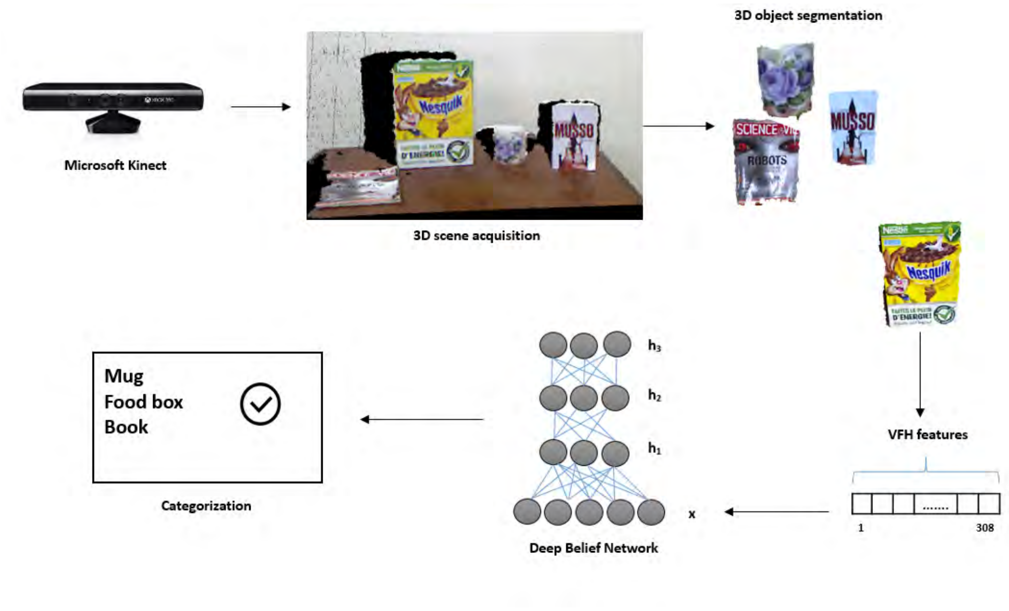


Figure 5.1: Overview of 3D global categorization approach.

5.1.1 3D scene segmentation

Over the past few years, various methods have been suggested in the literature for 3D point cloud segmentation. They are generally subdivided into five classes [174]: region-based methods, edges based methods, model-based methods, graph-based methods, and attributes based methods.

1. Region-based methods [188, 248] utilize the neighborhood information in order to combine nearly points which share common properties to obtain separated regions, and consequently obtain dissimilarity between different regions.

2. Edge-based methods [206] detect the edge of numerous regions in the point cloud to obtain segmented regions. These methods aim to locate the points that have a rapid change in intensity.

3. Model-based methods [210, 238] use the geometric primitive shapes such as a cylinder, a cone, a plane, and a sphere for grouping points. Those which have the same mathematical representation are grouped as one segment.

4. Graph-based methods [65] represent the point cloud as the graph. Each vertex corresponds to a point in the data, whereas the edges connect certain pairs of neighboring points.

5. Attributes based methods [25, 272] are based on clustering attributes of point cloud data. These methods include two separate phases. The first phase is attribute computation, in the second phase, point clouds will be clustered based on the computed attributes.

In our work, we used the Euclidean Cluster Extraction method presented in [198] to segment our 3D scenes. Rusu *et al.* provided a scene segmentation approach that consists of two major steps: (i) surface segmentation, and (ii) object segmentation. In their research scenario, the authors are most interested in the segmentation of surfaces which may contain tables as horizontal planes that can support objects on them. The reason for this segmentation can be more understood in the context of the final goals of the applications. A simple example concerns the problem of grasping objects with a mobile robot in the kitchen scene. In such a scenario, the surface is usually tables, a kitchen trolley, a kitchen counter, or kitchen cabinets. While the objects are movable such as small kitchen utensils, dishware, and food products, in the sense that the robot is able to manipulate them. The first challenge is how to find the plane areas of the kitchen where objects could be placed for grasping. Using the contextual knowledge, the robot knows that planar horizontal areas can support objects, and will proceed in searching for them. The method will make use of a Random Sample Consensus (RANSAC) method to speed up the search and to generate model hypotheses. While the model to be found represents a plane, and the three unique non-collinear points define a plane, the algorithm follows these steps:

1. select randomly three non-collinear unique points (p_i, p_j, p_k) from cloud model P;

2. compute the model coefficients from the three points $(ax + by + cz + d = 0)$;

3. compute the distances from all $p \in \mathrm{P}$ to the plane model (a, b, c, d);

4. count the number of points $p^* \in \mathrm{P}$ whose distance d to the plane model falls between $0 \leq |d| \leq |d_t|$, with d_t is a user-specified distance threshold.

Step 4 is one way of "scoring" a specific model. Each set of points p^* is stored, and the above steps are repeated for k iterations. After the algorithm is terminated, the set with the largest number of points (i.e., inliers) is selected as the support for the best planar model found.

After the selection of a planar model, the next major step is the segmentation of objects. First, the system requires to differentiate between object point cluster and another point cluster. Mathematically, a cluster is defined as follows. Let $\mathrm{O}_i = \{p_i \in \mathrm{P}\}$ be a distinct point cluster from $\mathrm{O}_j = \{p_j \in \mathrm{P}\}$ if:

$$min||p_i - p_j||_2 \geq d_{th} \tag{5.1}$$

With d_{th} represents a maximum distance threshold. Equation 5.1 states that if the minimum distance between a set of points $p_i \in \mathrm{P}$ and another set $p_j \in \mathrm{P}$ is larger than a given distance value d_{th}, then the points in p_i are set to belong to a point cluster O_i and the ones in p_j

to another distinct point cluster O_j. In this method, a maximum and minimum cluster size can be set to avoid under or over-segmentation. To handle these problems, one solution is to make use of approximate nearest neighbors queries using kd-tree representations. In summary, the Euclidean Cluster Extraction algorithm which constructs a set of separated Euclidean object clusters is computed as follows:

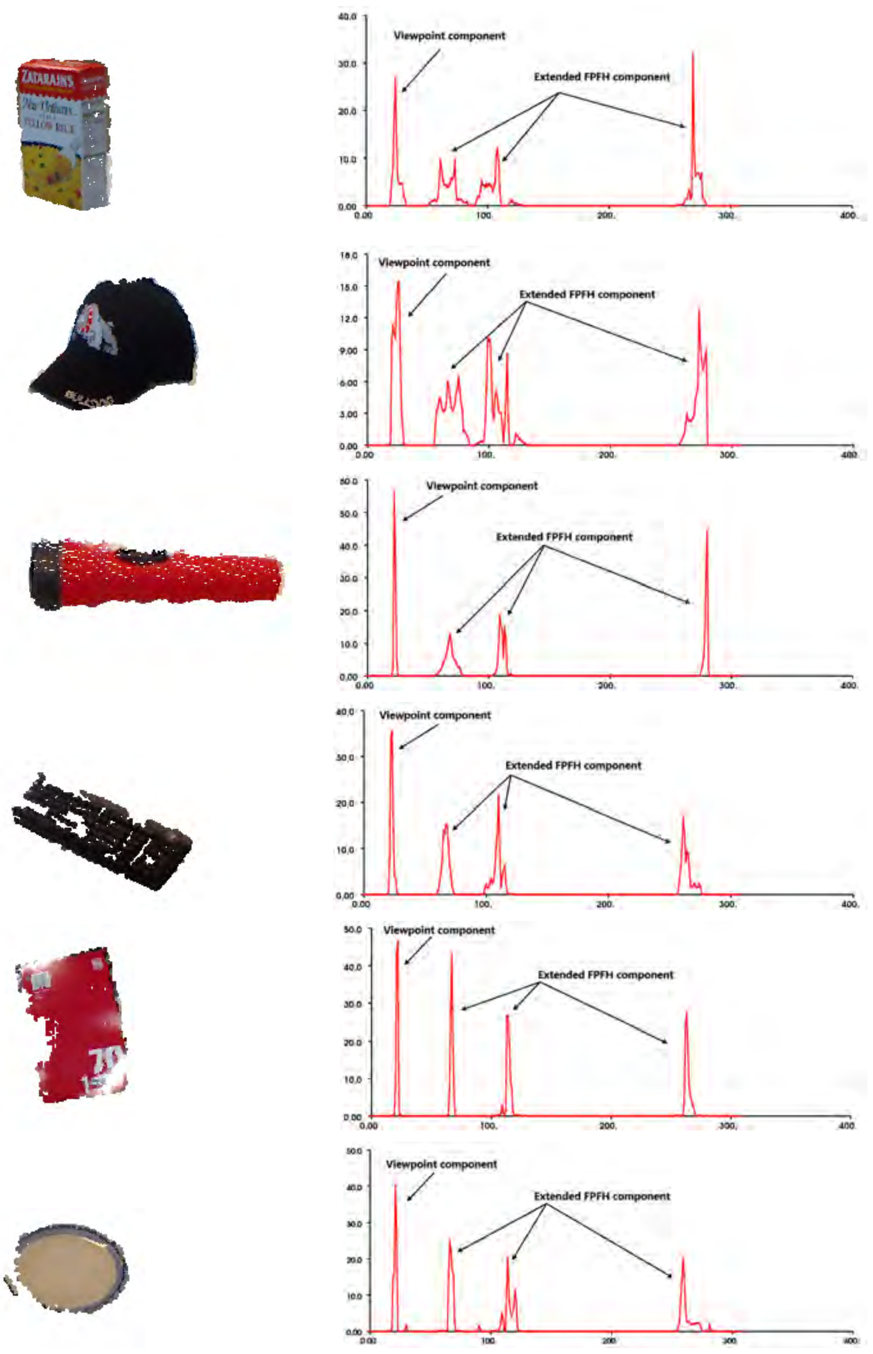
1. create a kd-tree representation for the input point cloud P;
2. set up an empty list of clusters C, and a queue of the points which need to be checked Q;
3. then for every point $p_i \in P$, perform the following steps:
    - add p_i to the current queue Q;
    - for every point $p_i \in Q$ do:
        - search for the set P_i^k of point neighbors of p_i in a sphere with radius $r < d_{th}$;
        - for every neighbor $p_i^k \in P_i^k$, check if the point has already been processed, otherwise add it to Q;
    - when the list of all points in Q has been processed, add Q to the list of clusters C and reset Q to an empty list.
4. the algorithm terminates when all points $p_i \in P$ have been processed and are now part of the list of point clusters C.

5.1.2 Feature description

The role of feature description is to encode the properties of the object in a robust, stable, and discriminating manner in order to categorize or recognize objects based upon their representation in the feature space. The methods adopted for feature description can be divided into two main families: local and global. Global descriptors can be regarded as a more natural choice since they describe the whole object with a single feature vector. This yields global features more discriminating by taking into account the entire object geometry. VFH is an extension to the FPFH and represents the most widely used descriptor for global approaches. The viewpoint component is computed by collecting a histogram of the angles that the viewpoint direction makes with each normal. The second component measures the relative (α, θ, ϕ) angles which are measured between the viewpoint direction at the central point and each of the normals on the surface as described in Fast Point Feature Histograms (FPFH) descriptors. More details about this descriptor are provided in Section 3.2.8. Figure 5.2 presents the combination of both a viewpoint direction component as well as a surface shape component comprised of an extended FPFH.

5.1.3 Feature categorization

Deep Belief Network (DBN) is a graphical model consisting of undirected networks at the top hidden layers and directed networks in the lower layers. The learning algorithm uses greedy layer-wise training by stacking RBMs which contain a hidden layer for modeling the probability distribution of visible variables. The idea of having multiple hidden layers is that the preceding hidden layer acts as the visible layer for the next hidden layer and thus the model can incrementally learn more complex features of data. There exist two types of DBN: Generative DBN (GDBN) and Discriminative DBN (DDBN). Both types use the greedy layer-wise



(a) 3D objects (b) VFH descriptor histograms

Figure 5.2: (a) 3D point clouds of object samples. (b) VFH descriptor of point clouds: the x-axis represents a number of histogram bin and y-axis represents a percentage of points falling in each bin.

algorithm. The terms generative and discriminative refer to the nature of RBMs used in each architecture. The first one is composed of GRBMs and adds a final layer of variables that represent the desired outputs then performs a purely discriminative fine-tuning phase using backpropagation. We refer to this architecture as "GDBN" or also "BP-DBN" (backpropagation DBN) (Section 2.2.3). While the second one consists of GRBMS in the first layers and DRBM in the last one. We refer to this architecture as "DDBN" (Section 2.2.3). Also, we used Gaussian-Bernoulli Restricted Boltzmann Machines (GBRBMs) (Section 2.2.2) which models real-valued inputs that are very appropriate to our VFH descriptors as input data. Finally, Contrastive Divergence (CD), Persistent Contrastive Divergence (PCD), and Free Energy in Persistent Contrastive Divergence (FEPCD) are used to train the GBRBMs (Section 2.2.2). CD is the most popular gradient approximation algorithm. CD initializes the Markov chain with training hidden units are computed. Once the method defines binary hidden unit states then the visible values are recalculated. Lastly, the probability of hidden unit instigations is calculated by means of hidden and visible unit's values [79]. As the CD sampling has a few drawbacks and is not precise, PCD method is proposed so that only the last chain state is used in the preceding update step [242]. Numerous insistent chains can be utilized in parallel during PCD sampling and can mention the present state as fantasy points in each of these chains. However, there is a blind chain selection and it's not necessary that the best one is always selected. Recently, Keyvanrad and Homayounpour [100] proposed a new sampling method that defines a standard for the goodness of chain. This method employs free energy as a measure to acquire the best samples from the generative model that are able to precisely calculate the gradient of log probability from training data.

5.2 Experimental results

The proposed object categorization approach was implemented with point cloud library PCL and deep learning toolbox. The RGBD camera used for the scene acquisition was a Microsoft Kinect sensor. To evaluate DBN architectures in the context of 3D object categorization, we tested two scenarios. In the first one, we used Washington RGBD object categories as training and testing data. Whereas, in the second scenario, we used Washington RGBD object categories only as training data. The testing data are represented by the segmented objects which are present in the 3D plane scenes.

5.2.1 Washington RGBD dataset

The effectiveness of the proposed categorization approach is evaluated using Washington dataset [111], which represents a large-scale multi-view object dataset collected using an RGBD camera. For experimental evaluation, we selected ten object categories that contain three object instances per category. Figure 5.3 illustrates different categories that are selected including a plate, an orange, a notebook, a mug, a keyboard, a food box, a flashlight, a cap, a bowl, and an apple. These categories show little, partial or strong similarity (e.g., food box, keyboard, and notebook contain flat parts, apple, and orange contain spherical parts, as well as bowl and cap, contain bulging surfaces). The point clouds are split into training and testing sets with average ratio 70% and 30% respectively per category.

5.2.2 RGBD scene segmentation

As described in Section 5.1.1, Euclidean cluster extraction algorithm consists of two major steps: (i) plane surface segmentation, and (ii) object segmentation. In the plane surface

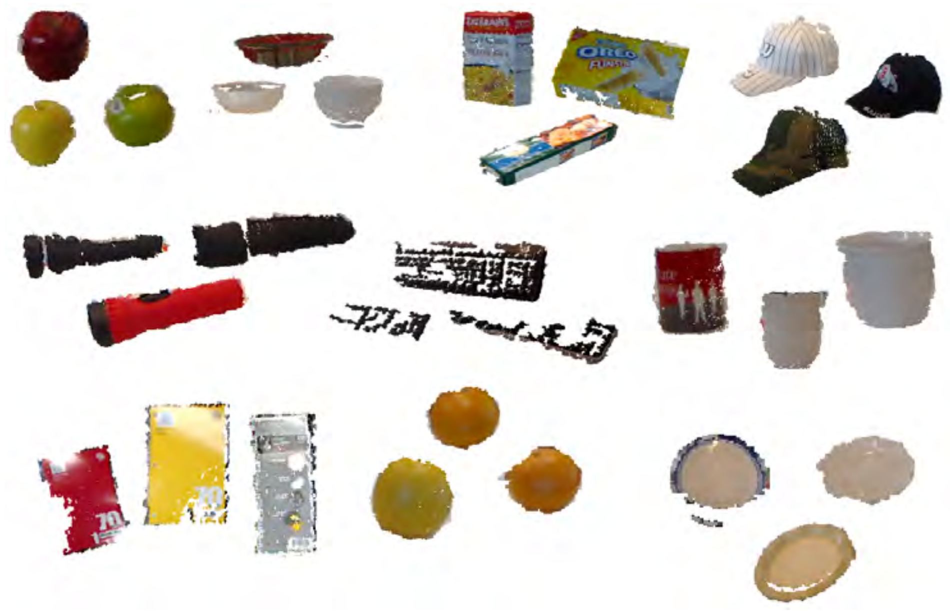


Figure 5.3: The category instances used in our experiments.

segmentation, we specify the distance threshold $d_{th} = 0.02m$ in order to determine how close a point must be to the surface to be considered as an inlier. Then, we used the RANSAC method with maximum iterations $k = 100$ as the robust estimator of choice. After this step, the algorithm performed the Euclidean cluster extraction by setting the right parameters for the segmentation. We carefully selected the right value for the cluster tolerance which is equal to $1.5cm$ to avoid either an actual object can be seen as multiple clusters or that multiple objects are seen as one cluster. We also imposed that the clusters found must have at least a minimum cluster size equal to 300 points and maximum cluster size equal to 25000 points.

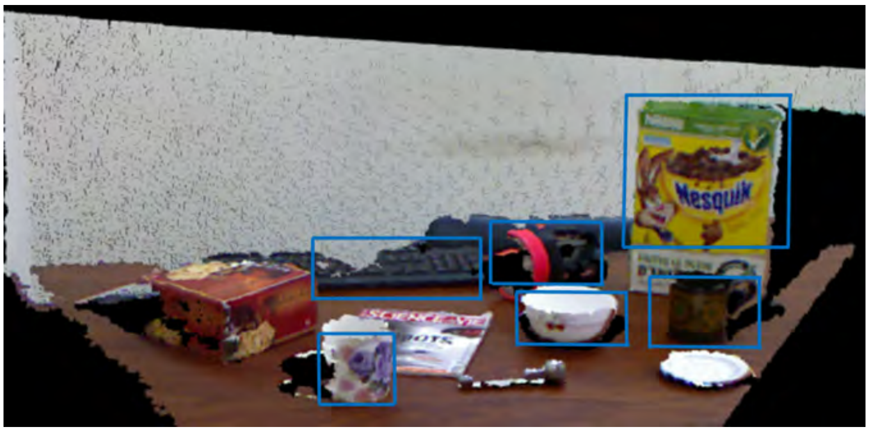


Figure 5.4: Planar model of our laboratory 3D scene.

Consider Figure 5.4 which shows a real-world indoor scene composed of several familiar objects with different shapes. Note that because the objects occlude one another, parts from different objects are intermingled. Blue boxes indicate the objects that are segmented from the cluttered scene. We can observe that the objects that are mingled in the planar surface model (i.e., a table) such as a plate, small flashlight, and a magazine are not segmented since they are considered by the algorithm as part of the planar surface model. Figure 5.5 shows the Euclidean clustering results for the points supported by the horizontal planar model.

The algorithm segments only some object parts which are captured by the Microsoft Kinect camera.

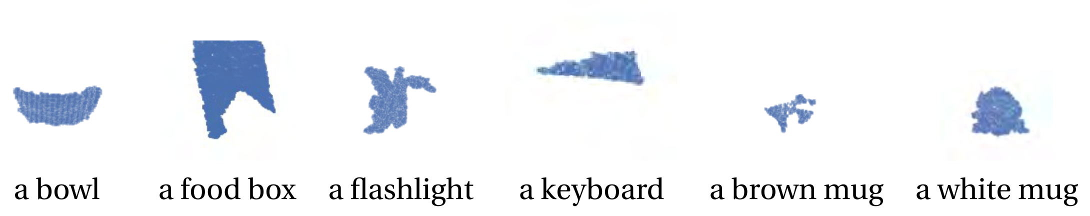


Figure 5.5: Euclidean clustering results for the points supported by the horizontal planar model presented in Figure 5.4.

According to Figure 5.6, the algorithm segments the intermingled objects food box and bowl into one cluster object. Consequently, this segmentation causes some misclassification in the categorization task. In section 5.2.6, DDBN confuses in the classification results a bowl with a food box objects. This result is obvious since, after segmentation, the cluster object combines food box as well as bowl parts into a single object.

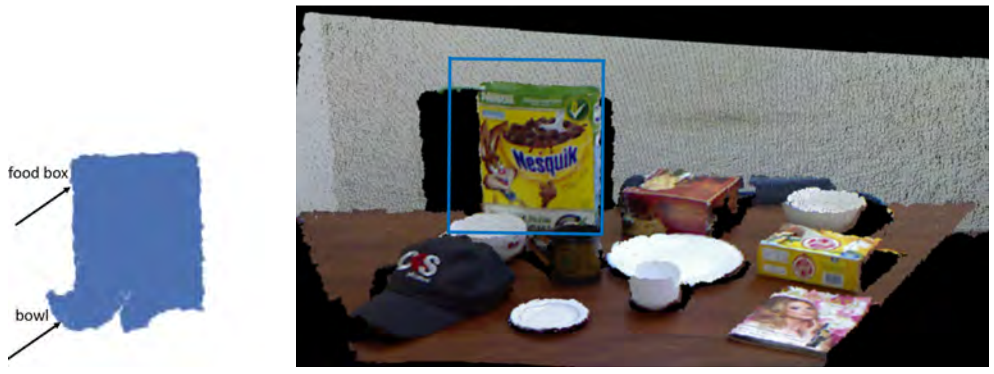


Figure 5.6: Segmentation of intermingled objects into one object from our laboratory.

According to the results of Figure 5.7, the segmentation algorithm can not segment all the objects present in the scene. Some objects which don't have an important height are confused with the planar surface model and are not selected as segmentation candidates. Whereas, bowl shape is accurately segmented from the scene.

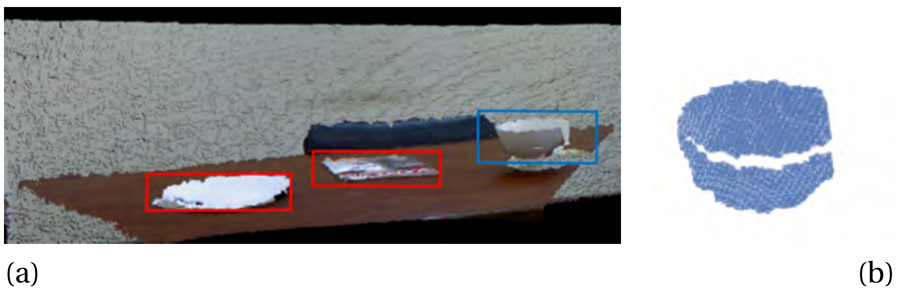


Figure 5.7: (a) Segmentation results. Blue box represents the object which is segmented. Red boxes represent objects which are not segmented by the algorithm. (b) The segmented object bowl.

5.2.3 Evaluation metrics

Ideally, after training a neural network to perform prediction, we have a testing data with which we evaluate the trained network. For every prediction, the true value of the testing data as predicted by the network is known. And also, the true value of the actual output called *target* is known. The problem now is how to compare the predicted value with the target one. Most neural network training algorithms aim to minimize the mean square error of the output(s). Naturally, during training, this is done in terms of the scaled data that the network actually sees. For our proposes, we will assume that the network's outputs have been unscaled so that we are working with the real problem data. The Mean Square Error (MSE) is computed by finding the difference between the desired target output and the attained one, squaring it, and summing across all trials n. Then, the sum is divided by n to get a mean value. The MSE for n trials is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n-1}(t_i - o_i)^2 \tag{5.2}$$

With: t_i is the target value of the prediction for trial i, and o_i is the value obtained by the network. MSE represents the simple error measure to minimize.

5.2.4 Experimental setup

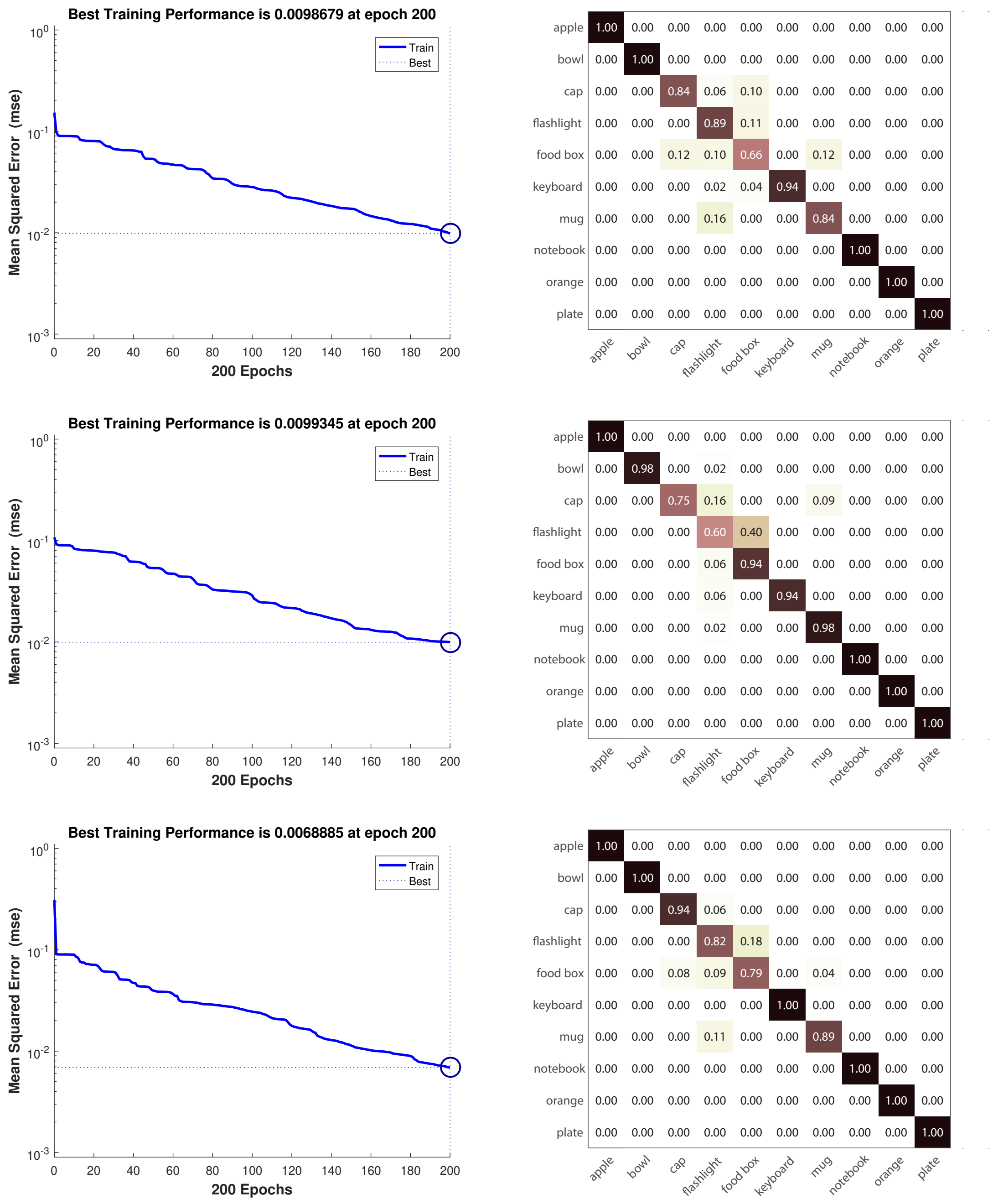
The algorithm is implemented on a Xeon(R) 3.50 GHz CPU 32 GB RAM and K2000 Nvidia card on Ubuntu 14.04. The code is written in Matlab and C++ using PCL library. We set a GDBN/DDBN architectures with one visible layer that contains the VFH descriptors of all training sets, as well as three hidden layers in order to define a 308-300-300-1500 GDBN and DDBN structures according to the experiments. Then, we train the weights of each layer separately with the fixed number of epochs equal to 200. Table 5.1 summarizes GDBN/DDBN characteristics which are used in our experiments.

Characteristics	**Values**
Hidden layers	3
Hidden layer units	300-300-1500
Learn rates	0.0001
Epochs	200
Input layer units	size of VFH descriptor (308)

Table 5.1: GDBN/DDBN characteristics that are used in our experiments.

5.2.5 Evaluation of generative model: GDBN

GDBN aims at allowing each GRBM model in the sequence to receive a different representation of the data. In other words, after GRBM is trained, the activity values of its hidden units are used as the training data for learning a higher-level GRBM. As a comparison, we evaluate the training process of GDBN using CD, PCD or FEPCD training methods. The best training performance indicates the iterations at which the validation performance reaches a minimum mean squared normalized error (MSE). As shown in Figure 5.8, the minimum MSE achieved the value 0.0068885 and was reached at epoch 200 with FEPCD sampling method. The figure shows also the experimental results of GDBN classification obtained on Washington RGBD dataset. Confusion matrices contain information about the right classifications and misclassifications done by a GDBN using different sampling methods. In CD and PCD training, GDBN misclassifies different categories including a cap, a flashlight, a food box, a mug, a keyboard, and a bowl. Contrary to CD and PCD training, GDBN using FEPCD training confuses only between a flashlight, a food box, a cap, and a mug. Figure 5.9 illustrates the minimum MSE value 0.019209 that was reached at epoch 200 with FEPCD sampling method. This result also ensures that FEPCD training method is more appropriate relating to the other training methods such as CD or PCD. Table 5.2 shows that the classification error decreases more with the FEPCD training method compared to the other training methods. We can also remark that FEPCD training method presents the best accuracy value for both datasets. Moreover, Table 5.2 shows that in real environments, 3D categorization accuracy obtained is 93.56%.



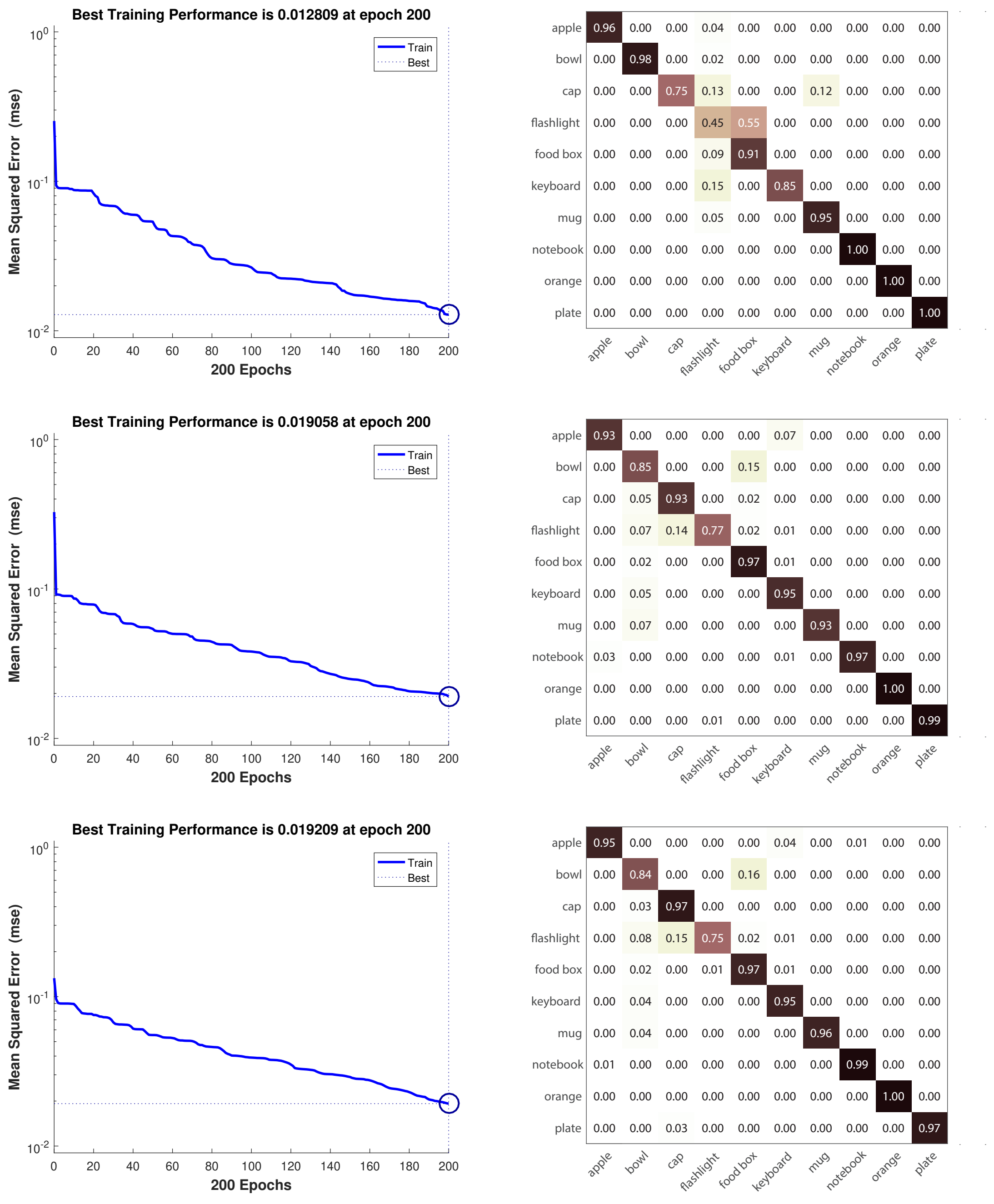
	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bowl	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.00	0.84	0.06	0.10	0.00	0.00	0.00	0.00	0.00
flashlight	0.00	0.00	0.00	0.89	0.11	0.00	0.00	0.00	0.00	0.00
food box	0.00	0.00	0.12	0.10	0.66	0.00	0.12	0.00	0.00	0.00
keyboard	0.00	0.00	0.00	0.02	0.04	0.94	0.00	0.00	0.00	0.00
mug	0.00	0.00	0.00	0.16	0.00	0.00	0.84	0.00	0.00	0.00
notebook	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bowl	0.00	0.98	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.00	0.75	0.16	0.00	0.00	0.09	0.00	0.00	0.00
flashlight	0.00	0.00	0.00	0.60	0.40	0.00	0.00	0.00	0.00	0.00
food box	0.00	0.00	0.00	0.06	0.94	0.00	0.00	0.00	0.00	0.00
keyboard	0.00	0.00	0.00	0.06	0.00	0.94	0.00	0.00	0.00	0.00
mug	0.00	0.00	0.00	0.02	0.00	0.00	0.98	0.00	0.00	0.00
notebook	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bowl	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.00	0.94	0.06	0.00	0.00	0.00	0.00	0.00	0.00
flashlight	0.00	0.00	0.00	0.82	0.18	0.00	0.00	0.00	0.00	0.00
food box	0.00	0.00	0.08	0.09	0.79	0.00	0.04	0.00	0.00	0.00
keyboard	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
mug	0.00	0.00	0.00	0.11	0.00	0.00	0.89	0.00	0.00	0.00
notebook	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 5.8: Left: best training performance of 308-300-300-1500 GDBN structure on Washington RGBD dataset. Right: confusion matrices of GDBN using Washington RGBD data as training and testing. Each line from top to down corresponds to CD, PCD, and FEPCD sampling respectively.

Figure 5.10 shows the confused objects which have some similar shape parts. For in-



	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	0.96	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
bowl	0.00	0.98	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.00	0.75	0.13	0.00	0.00	0.12	0.00	0.00	0.00
flashlight	0.00	0.00	0.00	0.45	0.55	0.00	0.00	0.00	0.00	0.00
food box	0.00	0.00	0.00	0.09	0.91	0.00	0.00	0.00	0.00	0.00
keyboard	0.00	0.00	0.00	0.15	0.00	0.85	0.00	0.00	0.00	0.00
mug	0.00	0.00	0.00	0.05	0.00	0.00	0.95	0.00	0.00	0.00
notebook	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	0.93	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00
bowl	0.00	0.85	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.05	0.93	0.00	0.02	0.00	0.00	0.00	0.00	0.00
flashlight	0.00	0.07	0.14	0.77	0.02	0.01	0.00	0.00	0.00	0.00
food box	0.00	0.02	0.00	0.00	0.97	0.01	0.00	0.00	0.00	0.00
keyboard	0.00	0.05	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00
mug	0.00	0.07	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.00
notebook	0.03	0.00	0.00	0.00	0.00	0.01	0.00	0.97	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.99

	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	0.95	0.00	0.00	0.00	0.00	0.04	0.00	0.01	0.00	0.00
bowl	0.00	0.84	0.00	0.00	0.16	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.03	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00
flashlight	0.00	0.08	0.15	0.75	0.02	0.01	0.00	0.00	0.00	0.00
food box	0.00	0.02	0.00	0.01	0.97	0.01	0.00	0.00	0.00	0.00
keyboard	0.00	0.04	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00
mug	0.00	0.04	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00
notebook	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.97

Figure 5.9: Left: best training performance of 308-300-300-1500 GDBN structure on our data. Right: the confusion matrices of GDBN using Washington RGBD data as training and our data as testing. Each line from top to down corresponds to CD, PCD, and FEPCD sampling respectively.

stance, the pair objects which are confused are a cap with a flashlight, a food box with a flashlight, a mug with a flashlight, a keyboard with a flashlight, and a flashlight with a cap.

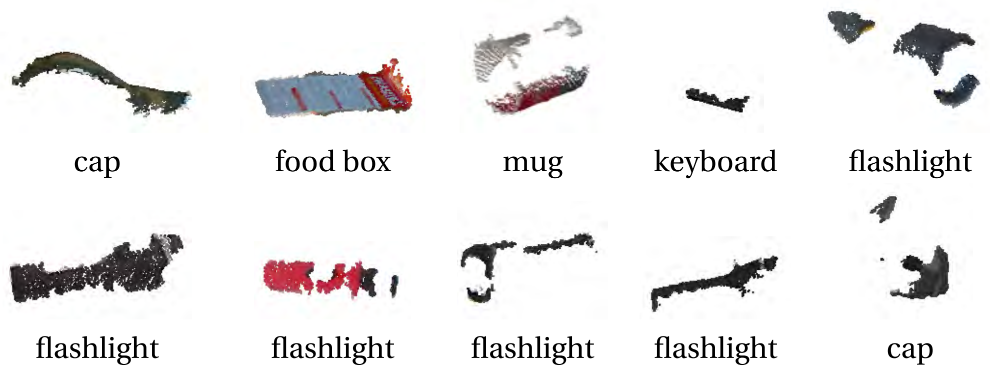


Figure 5.10: Confused objects which have some similar parts.

	Washington data		Our data	
	Error	Accuracy	Error	Accuracy
CD-GDBN	0.086	91.79%	0.126	88.53%
PCD-GDBN	0.078	91.87%	0.0793	92.81%
FEPCD-GDBN	**0.056**	**94.38%**	**0.0757**	**93.56%**

Table 5.2: Classification error and accuracy rate in GDBN experiments.

5.2.6 Evaluation of discriminative model: DDBN

The approach trains RBMs one after another and uses their training data resulting in the training stage in the next RBM using CD, PCD or FEPCD training methods. The last layer trains a joint density model with a discriminative RBM. We use the backpropagation technique through the whole classifier to fine-tune the weights in order to optimize the classification result. As shown in Figure 5.11, the best training performance 0.006076 is obtained with FEPCD training. Figure 5.11 shows also the confusion matrixes across all 10 categories. Most model's results of FEPCD training are very reasonable showing that DDBN can misclassify only a cap, a mug, a flashlight, and a food box objects. Figure 5.12 shows that the best training performance 0.018663 is obtained with FEPCD training. According to Table 5.3, FEPCD training method is more appropriate relating to the other training methods (i.e., CD and PCD). These results show that the gradient is computed using better and more accurate samples. The best result was obtained with DDBN using FEPCD method with 96.43% accuracy. While the FEPCD classification error is decreased to 0.036 in the experiments which are conducted on Washington RGBD dataset and to 0.0715 in our real 3D scenes.

In general, the use of PCD training is better than CD training, and FEPCD outperforms PCD training. This result is pertinent since FEPCD uses free energy as a criterion for the goodness of a chain in order to obtain elite samples from the generative model that can more accurately compute the gradient of the log probability of training data. Also, discriminative training contrary to generative one holds the promise of learning powerful end to end systems have given enough labeled training data. In summary, we can conclude that DDBN using FEPCD training can improve the performance of 3D categorization.

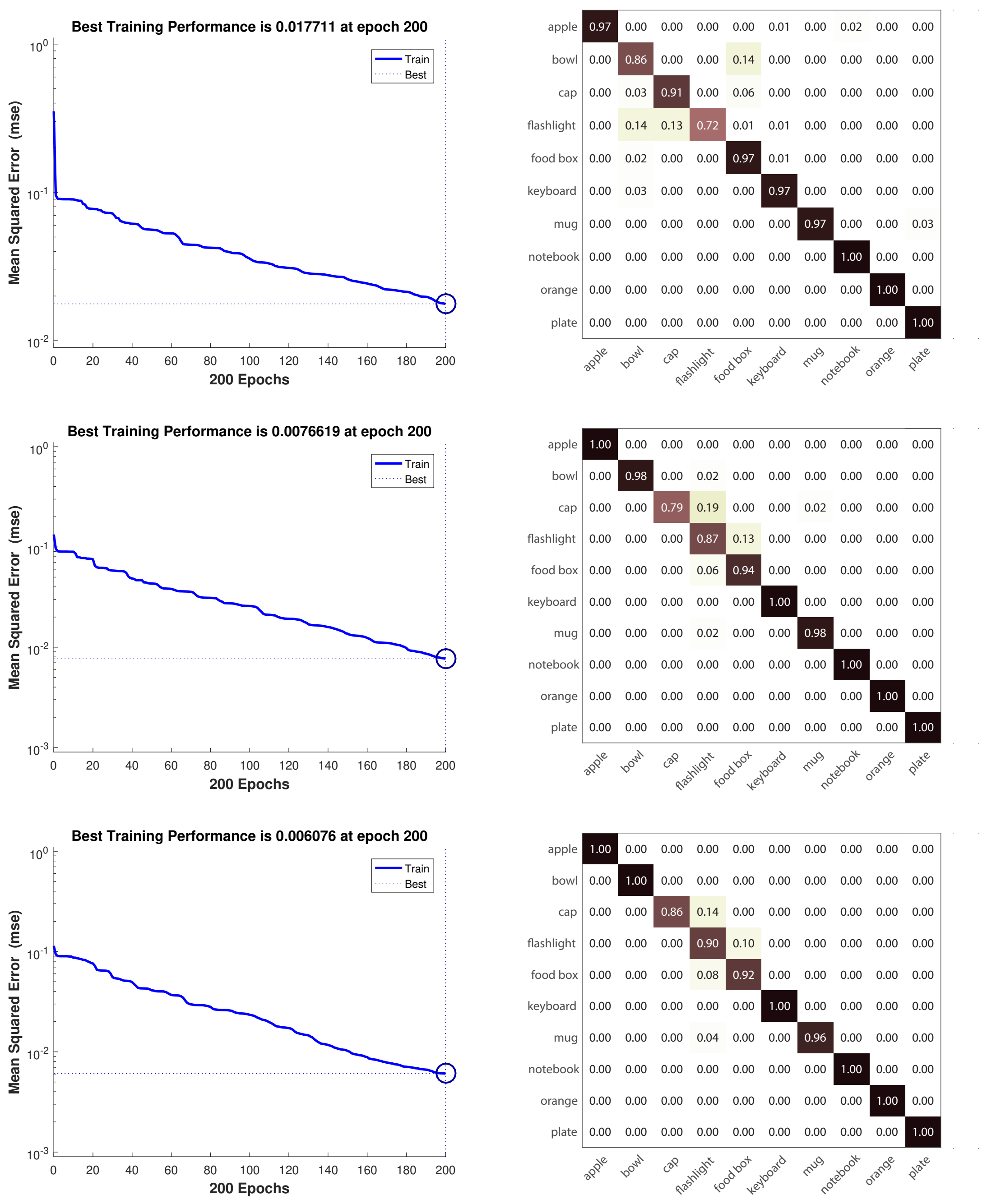
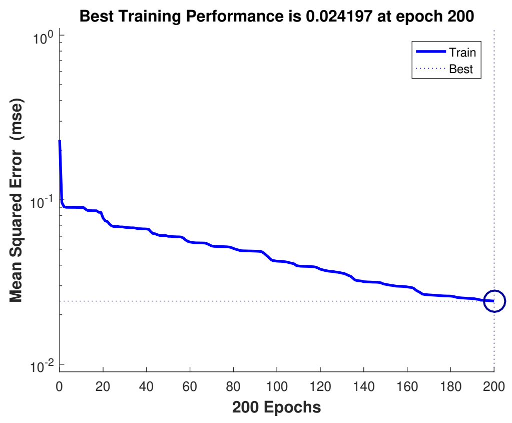
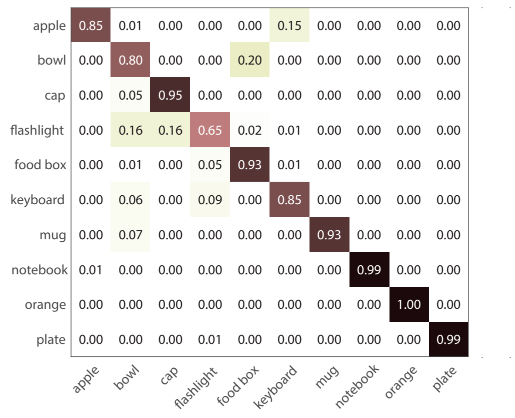


Figure 5.11: Left: best training performance of 308-300-300-1500 DDBN structure on Washington RGBD dataset. Right: confusion matrices of DDBN using Washington RGBD data as training and testing. Each line from top to down corresponds to CD, PCD, and FEPCD sampling respectively.

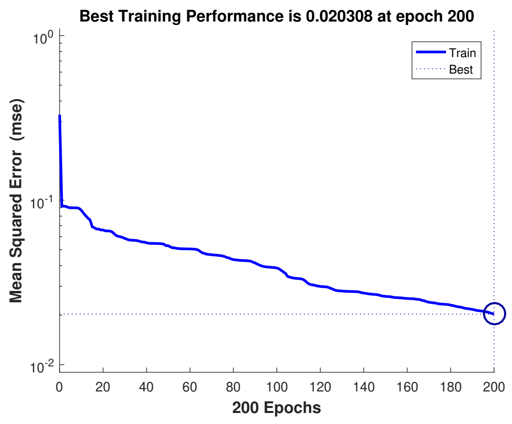
	Washington data		Our data	
	Error	Accuracy	Error	Accuracy
CD-DDBN	0.0751	93.81%	0.1236	89.14%
PCD-DDBN	0.046	95.59%	0.0836	92.60%
FEPCD-DDBN	**0.036**	**96.43%**	**0.0715**	**94.26%**

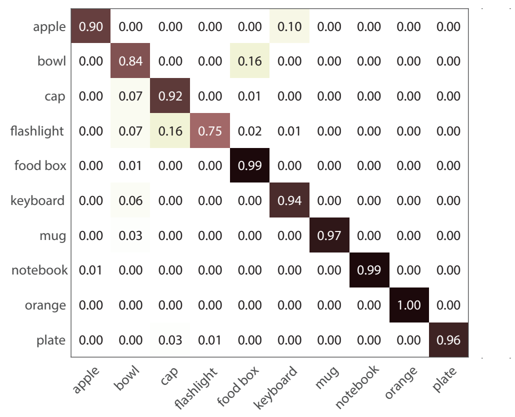
Table 5.3: Classification error and accuracy rate in DDBN experiments.



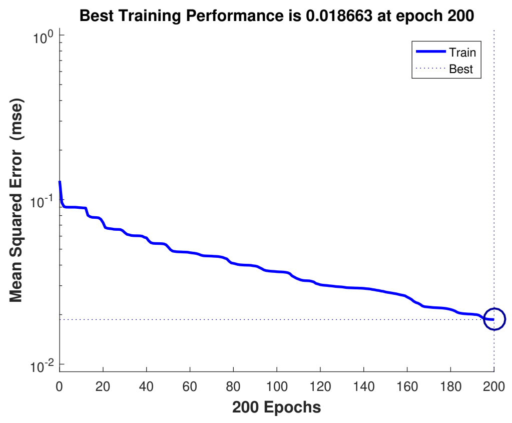


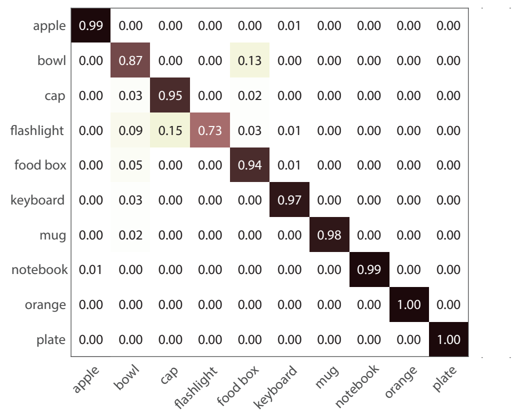
	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	0.85	0.01	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00
bowl	0.00	0.80	0.00	0.00	0.20	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.05	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00
flashlight	0.00	0.16	0.16	0.65	0.02	0.01	0.00	0.00	0.00	0.00
food box	0.00	0.01	0.00	0.05	0.93	0.01	0.00	0.00	0.00	0.00
keyboard	0.00	0.06	0.00	0.09	0.00	0.85	0.00	0.00	0.00	0.00
mug	0.00	0.07	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.00
notebook	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.99





	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	0.90	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00
bowl	0.00	0.84	0.00	0.00	0.16	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.07	0.92	0.00	0.01	0.00	0.00	0.00	0.00	0.00
flashlight	0.00	0.07	0.16	0.75	0.02	0.01	0.00	0.00	0.00	0.00
food box	0.00	0.01	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00
keyboard	0.00	0.06	0.00	0.00	0.00	0.94	0.00	0.00	0.00	0.00
mug	0.00	0.03	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00
notebook	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.03	0.01	0.00	0.00	0.00	0.00	0.00	0.96





	apple	bowl	cap	flashlight	food box	keyboard	mug	notebook	orange	plate
apple	0.99	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
bowl	0.00	0.87	0.00	0.00	0.13	0.00	0.00	0.00	0.00	0.00
cap	0.00	0.03	0.95	0.00	0.02	0.00	0.00	0.00	0.00	0.00
flashlight	0.00	0.09	0.15	0.73	0.03	0.01	0.00	0.00	0.00	0.00
food box	0.00	0.05	0.00	0.00	0.94	0.01	0.00	0.00	0.00	0.00
keyboard	0.00	0.03	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00
mug	0.00	0.02	0.00	0.00	0.00	0.00	0.98	0.00	0.00	0.00
notebook	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
plate	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 5.12: Left: best training performance of 308-300-300-1500 DDBN structure on our data. Right: the confusion matrices of DDBN using Washington RGBD data as training and our data as testing. Each line from top to down corresponds to CD, PCD, and FEPCD sampling respectively.

5.2.7 Comparison to other methods

In this subsection, we compare our approach to related state-of-the-art approaches. Table 5.4 shows the main accuracy values and compares our 3D categorization to the published results [50, 111, 144, 211, 230, 276]. Lai *et al.* [111] extracted a set of features that captures the shape of the object view using Spin Images, and another set which captures the visual appearance using SIFT descriptors. These features are extracted separately from both depth and RGB images. In contrast, we extract the geometric features from a single point cloud using only the VFH descriptor. Schwarz *et al.* [211] used both colorizing depth and RGB images that are processed independently by a CNN. CNN features are then learned using SVM classifier in order to successively determine the category, instance, and pose. Eitel *et al.* [50] presented two separate CNN processing streams for RGBD object recognition. RGB and colorized depth images consist of five convolutional layers and two fully-connected layers. Both streams are processed separately through several layers and converge into one fully-connected layer and a softmax layer for the classification task. Madai *et al.* [144] reinvestigated Deep Convolutional Neural Networks (DCNNs) for RGBD object recognition. They proposed a new method for depth colorization based on surface normals, which colorized the surface normals for every pixel and computed the gradients in a horizontal direction (x-axis) and vertical direction (y-axis) through the Sobel operator. The authors defined two 3D vectors a and b in direction of the z-axis in order to calculate the surface normal n. As n has 3 dimensions, the authors mapped each of the three values of the surface normal to a corresponding RGB channel. Zaki *et al.* [276] provided an effective recognition framework based upon a pre-trained CNN on RGB data as feature extractor for both depth and color channels as well as point clouds. They defined a new Hypercube representation that encodes activations of all convolutional layers in order to preserve spatially discriminative features, in addition to the semantically-informative global features extracted from the fully connected layer. For multiscale feature extraction, they devised a coarse-to-fine scheme based on pyramidal re-sampling of the convolutional feature maps. Then, they used a spatial pyramid pooling scheme at each pyramid level before feature concatenation to produce a compact feature representation. The feature vectors from these pyramids are max pooled to produce a single Hypercube Pyramid descriptor. In the last step, these feature vectors from multiple levels and modalities are given as input to Extreme Learning Machine classifiers to perform object, category, and instance recognition. A recent work by Sun *et al.* [230] designed a PCA–CCA network method for RGBD object recognition. It is composed of PCA filter layer, CCA filter layer, binary hashing, and block-wise histograms. In the first layer, principal component analysis filters (PCA filters) are learned separately for RGB and depth in order to extract the most discriminative features in both modalities. Then, in the second layer, the CCA method generated the filters for the RGB and depth components, by maximizing the correlation between the two projected sets of variables. In this way, different characteristics of the RGB and depth modalities as well as the correlation between them are considered by the network. In our approach, we use VFH features for training GDBN/DDBN with three hidden layers that model a deep network architecture. The results show also that our recognition pipeline with DDBN architecture and FEPCD training works perfectly with the accuracy rate of 96.43% and outperforms all methods that are mentioned in the state-of-the-art.

Methods	**Accuracy rates**
CNN [211]	89.4%
kSVM [111]	83.8%
Eitel *et al.* [50]	91.3%
Madai *et al.* [144]	94%
Hypercube pyramid [276]	91.1%
PCA-CCA [230]	91.7%
DDBN-CD	93.81%
DDBN-PCD	95.59%
DDBN-FEPCD	**96.43%**

Table 5.4: The accuracy rates on Washington RGBD dataset compared with the state-of-the-art methods.

5.3 Conclusion

In this chapter, we focused on 3D object categorization using geometric features extracted from Viewpoint Feature Histogram (VFH) descriptor and learned with both Generative and Discriminative Deep Belief Network (GDBN/DDBN) architectures using CD, PCD, and FEPCD training methods. GDBN is the probabilistic model with many Restricted Boltzmann Machines (RBMs) which are trained sequentially. On the other hand, DDBN is constructed from the Discriminative Restricted Boltzmann Machine (DRBM) which is based on RBM and the joint distribution model. The experimental results using DDBN with FEPCD training method are encouraging, especially that our approach is able to classify 3D objects under different views. In future work, we will attempt to embed our algorithm in our robot TurtleBot2 in order to grasp the real-world objects.

Part II

Scene classification

Chapter 6

Scene Classification: literature review

"Try to forget what objects you have before you - a tree, a house, a field, or whatever. Merely think, 'Here is a little square of blue, here an oblong of pink, here a streak of yellow,' and paint it just as it looks to you, the exact color and shape, until it gives you your own impression of the scene before you."

Claude Monet

Contents

SCENE classification problem is one of the hardest challenges in computer vision. Given an arbitrary image, the goal is to describe what type of semantic scene it depicts. Despite the recent advances in computer vision, the human-like visual perception still perform better than the best available vision systems in scene classification. As such, it is important to examine the low-level computational architecture according to which human vision is organized. Also, advances in 3D sensing and the availability of low-cost 3D sensors like Microsoft Kinect have made it possible to record depth information along with RGB images. As result, the use of RGBD data for scene classification task has recorded tremendous attention in the last years. In this chapter, some of the important concepts in visual attention are introduced from a psychophysical neurobiological perspective. Then, the existing state-of-art approaches to visual perception in computer vision is introduced.

6.1 Visual attention modeling

Attention is a concept introduced in cognitive psychology, which refers to how we actively process specific information in our environment. According to psychologist and philosopher William James [88], attention "is the taking possession by the mind, in clear and vivid form, of one out of what may seem several simultaneously possible objects or trains of thought. It implies withdrawal from some things in order to deal effectively with others." Anderson defined [8] attention as "the behavioral and cognitive process of selectively concentrating on a discrete aspect of information, whether deemed subjective or objective, while ignoring other perceivable information." Visual attention is one type of attention where only part of the visual field is "processed" at any moment. In this section, some of the important concepts in visual attention are briefly introduced from a psychophysical and neurobiological perspective in order to identify our work context.

What is visual attention?

Human believe that they have a good resolution and accurate representation of the entire visual field at all times and that they see everything in their view instantaneously. However, the experimental and biological evidence shows that this concept is not true. It has been shown that they are only able to spend attention and computational resources on small regions of the scene at a time. Visual attention can be defined as one's perception of one of the aspects of information via visual sensors, shown by gaze [1] or its imitation, and one's concentration on it.

In human, attention is facilitated by a retina that has evolved a high-resolution central fovea as well as a low-resolution periphery. The visual attention guides this anatomical structure to prominent regions of the scene to gather more detailed information, the main question is on the computational mechanisms underlying this guidance. Recently, many researchers in different scientific fields have been aimed towards answering this question (Figure 6.1). Neurophysiologists showed how neurons accommodate themselves to better identify objects of interest [250]. Psychologists studied behavioral correlates of visual attention such as inattentional blindness [221], change blindness [124], and attentional blink [189]. Computational neuroscientists built realistic neural network models in order to explain and stimulate attentional behaviors [193]. They also provided two distinct types of attentional mechanisms; bottom-up and top-down which are widely used in computer vision and robotic applications.

[1]Gaze is defined as coordinated motion of the eyes and head, which has often been used as a proxy for attention in natural behavior [75].

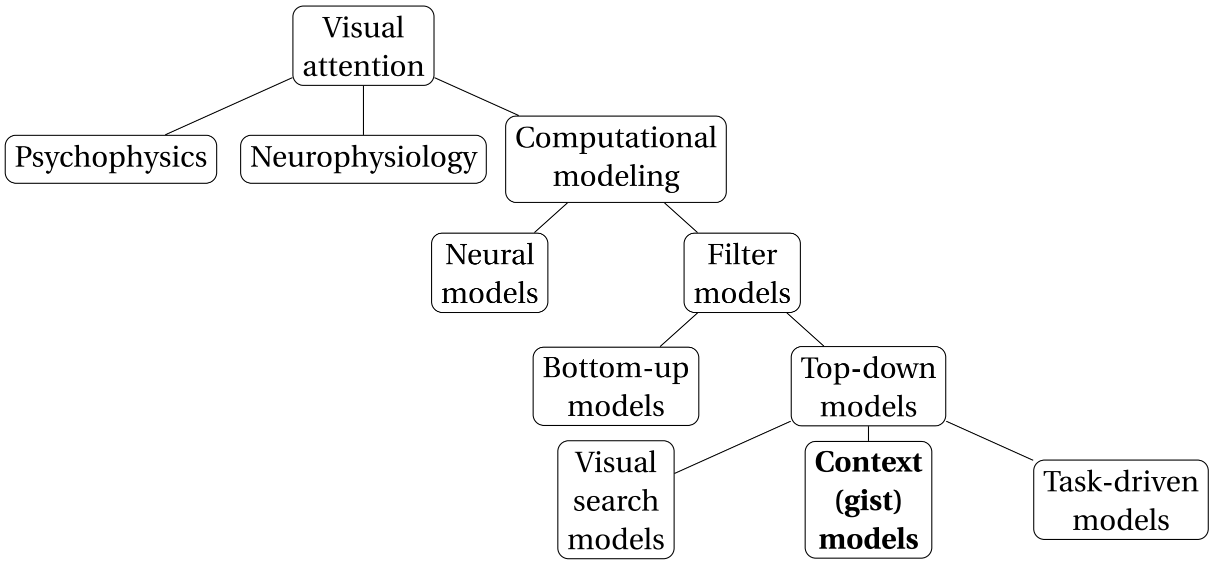


Figure 6.1: Taxonomy of visual attention studies [29].

Bottom-up visual attention

Bottom-up models are mainly based on characteristics of a visual scene (i.e., stimulus-driven). Regions of interest of visual scenes that attract our attention must be sufficiently distinctive with respect to the surrounding features. The bottom-up mechanism is also named automatic, exogenous, reflexive, or peripherally cued. This attentional mechanism is fast, involuntary, and most likely feedforward. A prototypical example of bottom-up attention is given in Figure 6.2 and looking at a scene with only one horizontal bar among several vertical bars where attention is immediately drawn to the horizontal bar.

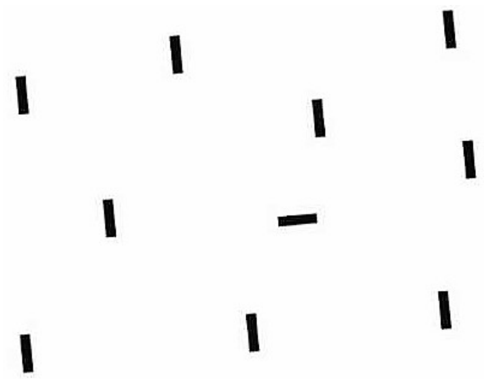


Figure 6.2: Example of bottom up visual attention.

Classic contrast-based approach was firstly introduced by Koch and Ulman [104] then it was redefined many times by researchers in their works. The Koch and Ulman model included computation of conspicuities for several features and combining them into a topographic map called a saliency map. The model is based on the Feature Integration Theory (FIT) [249] and used a Winner-Take-All (WTA) neural network in order to select salient regions. Then it employed an inhibition of return mechanism to allow the focus of attention to shift to the next most salient location. However, this model was only a description of the computational architecture and was not implemented at the time of publication. Milanese [163] introduced one of the earliest implementations of this model by using several features for computing saliency. He utilized two color opponencies (i.e., red-green and blue-yellow), 16 different orientations, local curvature, and intensity (where no color information is available). Then, he used center-surround mechanisms comparing the local feature values to their surroundings in order to compute the saliency per feature. These differences are then collected into a conspicuity map for each feature. Finally, all conspicuity maps are combined into one saliency map. Itti *et al.* [87] developed one of the best known visual attention systems named the Neuromorphic Vision Toolkit (NVT). NVT is based on the Koch-Ullman and Milanese models described above and have been used by many researcher groups in experiments and the development of visual attention systems. This work provided the first detailed description of the implementation of the ideas and approaches described by Koch and Ullman and Milanese in their works.

Feature extraction-based approach is a set of models, which aim to find individual elements of the scene, as well as the characteristics of these elements. Moreover, due to its nature, it can be seen as an ensemble of completely different interpretations. A feature can be seen as typical or descriptive for given problem context and is interpreted as any characteristic: orientation, form, the relative size of the element [278], or direction of movement or amplitude of repeated movements in video [37].

Neural network approach is widely developed in the last decades because of the evolution of convolutional deep neural networks [109]. This approach is based on one of two main principles. The first principle is that CNNs are very pertinent in terms of finding correlations and regularities both among evident and among hidden features of images. Second is that a deep neural network might be interpreted as an imitation of the multi-layer system of saccade programming in human's neural system that makes it more than just an abstract mathematical model.

Top-down visual attention

Top-down models (i.e., goal-driven) are determined by cognitive phenomena such as knowledge, reward, expectations, and current goals. This attentional mechanism is slow, task-driven, voluntary, and closed-loop. Yarbus [274] introduced the most famous example of top-down guidance by measuring eye trajectories in different tasks. He showed that eye movements depend on the current task and each of these tasks would produce a different pattern of eye movements. As shown in Figure 6.3, viewers (i.e., subjects) were asked to watch the same scene (a room with a family and an unexpected visitor entering the room) under different conditions: (a) no specific task, (b) estimate the wealth of the family, (c) estimate the ages of the people in the painting, (d) detect what the family had been doing before the arrival of the "unexpected visitor", (e) remember the clothes worn by the people, (f) remember the position of the objects and people in the room, and finally (g) estimate how long the "unexpected visitor" had been away from the family.

In general, top-down models have adopted three major sources in response to this question: how do we decide where to look? Some models investigated visual search in which attention is drawn toward features of a target object we are looking for. Some other models addressed the role of scene context or gist to constrain locations that we look at. While in principle, task demand models subsumed the other two factors, practically models have been focusing on each of them separately.

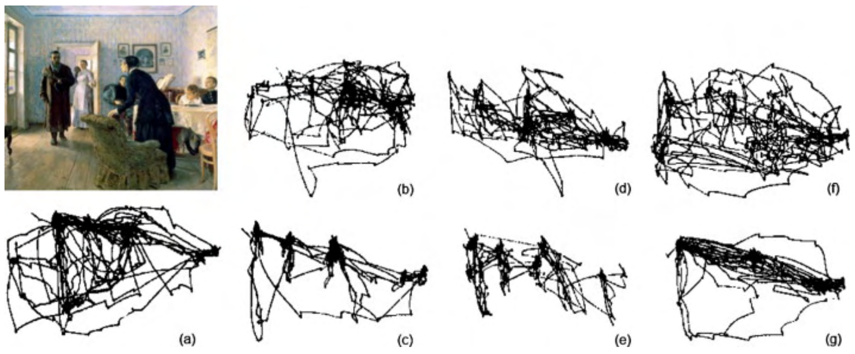


Figure 6.3: Eye trajectories measured by Yarbus by viewers carrying out different tasks [71].

Object features Consider an example of a red item search in the scene: attention is rapidly directed toward this target. Compare this with a more complex target object, such as a pedestrian in a natural scene, where although it is difficult to describe the target, there are still some features (e.g., upright form, round head, and straight body) to direct visual attention. The guided search theory [269] suggested that attention can be biased toward targets of interest by modulating the relative gains through which different features contribute to attention. To return to our prior example, when looking for a red object, a higher gain would be assigned to a red color.

Scene context An observer is able to report essential characteristics of a scene within a brief presentation of an image (i.e., 80 ms or less) [182]. This very rough representation of a scene, called *gist*, does not contain many details about distinctive objects but can provide sufficient information for whole scene discrimination (e.g., indoor vs. outdoor). It is important to notice that gist does not necessarily reveal the semantic category of a scene. More recently, gist representations have become increasingly popular in computer vision since they provide rich, global, and discriminative information useful for many applications such as search in the large-scale scene datasets, scene classification, and modeling top-down attention. It can thus be seen that research in this area has the potential to be very promising. In chapter 7, we will introduce in details gist representations, especially the one presented by Torralba [247].

Task demands Tasks have a strong influence on the deployment of attention. The finding of task-oriented studies is that the eyes are positioned at a point that is not the most visually salient but is the best for the spatiotemporal demands of the task that requires to be done. This line of investigation has been used in extended visuomotor tasks such as walking, driving, sports, and making tea or sandwiches [112, 113, 114]. The main result of all these investigations is that fixations are tightly linked in time to the evolution of the task. Ballard *et al.* [13] showed in their experiments that the information required for the task is acquired just prior to its use. This is called a *just-in-time* strategy, where observers acquire the specific information they need just at the point it is required in the task. Hayhoe and Ballard [75] showed that there is a strong relationship between visual cognition and eye movements when dealing with complex tasks. Subjects performing a visually-guided task were found to direct a majority of fixations toward task-relevant locations.

6.2 State-of-the-art

6.2.1 Visual attention-based approaches

Human vision uses the visual attention to select the interesting information relevant to the high-level cognitive processes such as video object segmentation [167], scene classification [217], visual tracking [264], and mobile robotic applications [253]. The concept of attention covers all factors that influence the selection mechanisms and is subdivided into two types: bottom-up attention and top-down attention. Bottom-up attention is mainly based on visual scene characteristics (i.e., stimulus-driven), while top-down attention (i.e., goal-driven) is defined by cognitive concepts such as current goals, knowledge, expectations, and reward. Siagian and Itti [217] described a scene recognition algorithm intended for mobile robotic applications. The model computed gist in a very inexpensive manner by using the same low-level visual front-end as the saliency model and used the neural network to classify outdoor and indoor scenes. This model is then extensively evaluated in three challenging outdoor environments across multiple days and times of days, where the dominating vegetation, shadows, and other ephemerous phenomena are expected to defeat landmark-based and region-based approaches. The experimental results showed that gist can reliably be extracted at very low computational cost, using very simple visual features shared with an attention system in an overall biologically-plausible framework. Also, the gist model can achieve reliable performance in outdoor localization for a walking human, with obvious application to autonomous mobile robotics. Joubert *et al.* [92] used a go/no-go rapid visual categorization task in which subjects had to respond as fast as possible when they saw a "man-made environment", or a "natural environment", that was flashed for only 26 ms. They evaluated the categorization task using sea, mountain, indoor, and street experiments and showed that the subjects were faster at completing the task. Also, "Man-made" and "natural" scenes were categorized with very high accuracy (both around 96%) and very short reaction times (i.e., median RT both around 390 ms). Fornoni and Caputo [58] complemented a traditional spatial-encoding scheme with a bottom-up approach provided in order to discover visual-structures regardless of their exact position in the scene. They made a well-known saliency operator and proposed a new saliency function, which directly employed the rich information encoded in the local descriptors to obtain the saliency map. Finally, they showed that the combination of the saliency-driven perceptual pooling with a simple spatial pooling scheme achieved state-of-the-art performances on two out of three publicly available scene recognition datasets: Indoor Scene Recognition (ISR) dataset, 15-Scenes dataset, and 8-Sports dataset. The results proved that the approach is effective in the indoor scenario while being also meaningful for other scene categorization tasks. Li *et al.* [126] argued that rapid visual categorization of new natural scenes needs very little or no focal attention. Such tasks that do not need attention appear to be carried out in the early stages of the visual system. Contrary to this common belief, they reported that subjects can rapidly detect vehicles or animals in briefly presented novel natural scenes while simultaneously performing another attentionally demanding task. By comparison, subjects are unable to discriminate bisected two-color disks from their mirror images under the same conditions. Finally, the authors concluded that some visual tasks associated with "high-level" cortical areas may proceed in the near absence of attention. Song and Tao [226] proposed a novel Biologically Inspired Feature Manifold (BIFM) framework for scene classification. BIFM consisted of three components: a new combination of popular Biologically Inspired Features (BIFs) for scene image representation, a novel discriminative subspace selection method, named Discriminative Geometry Preserving Projections (DGPP), for dimensionality reduction and a pairwise multi-class SVM classifier for classification. An image is presented based on a new BIF, which combines the intensity channel, the color channel, and the C1 unit of a color image. Then the high-dimensional BIF are projected to a low-dimensional space based on a novel manifold learning algorithm DGPP. DGPP precisely preserved both the intra-class geometry and inter-class discrimination. Empirical studies showed that DGPP outperformed the well-known dimensionality reduction algorithms, e.g., Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Locality Preserving Projections(LPP), Neighborhood Preserving Embedding (NPE), and Marginal Fisher Analysis (MFA), in the proposed BIFM for scene classification. Finally, the pairwise multi-class SVM is applied for final classification because of its good generalization ability for classification. The empirical studies based on the USC scene dataset demonstrated that BIFM improved the classification rates around 100% relatively and the training speed 60 times for different sites in comparing with the previous gist proposed by Siagian and Itti [217].

6.2.2 2D recognition and categorization approaches

Early approaches for indoor environment recognition relied on low-level features such as texture and color. Szummer and Picard [234] argued that classification of low-level image features leads to the extraction of high-level scene properties. They used the texture, color, and frequency features which are computed for the whole image and also for each sub-block of 4×4 image tessellation. The color features represent a color histogram that has 32 bins per channel. The three channels come from the Ohta color space. The choice of this color space is justified by the fact that the color channels are approximately decorrelated, which is adequate for computing per-channel histograms. The color axes of Ohta are the largest eigenvectors of the RGB space, found through PCA of a large set of natural images. The texture features are extracted from grayscale images using the Multiresolution Simultaneous Auto-Regressive model (MSAR). MSAR constructs the best linear predictor of a pixel based upon the noncausal neighborhood. The features represent the weights of the predictor. The authors used three different neighborhoods at 2,3 and 4. Then the weights are concatenated to yield a 15-dimensional vector. Finally, the extracted features are classified by means of the k-nearest neighbor method.

Color histograms of the acquired omnidirectional images and the k-nearest neighbor method for classification are introduced by Ulrich and Nourbakhsh [255]. Their approach is applied for topological localization in a mobile robot. The key module of their topological localization approach is the place recognition. They assumed that the previously learned set of omnidirectional images is representative of the locations. The goal of the place recognition is to determine the reference image that is most similar in appearance to the current image. Firstly, the place recognition function builds color histograms of the acquired input image. Then, for each color band, it determines the best match for each candidate location using Jeffrey divergence metric. For each candidate location and each color band, the distance between the reference histograms and the input histogram is thus computed using the Jeffrey divergence. For each candidate location, the minimum distance is then determined for each band. After that, each color band individually determined location vote with the smallest minimum matching distance. Finally, the six votes from the color bands are combined to classify the input image. Serrano *et al.* [213] introduced an improved approach to indoor/outdoor classification using a low complexity as well as a low-dimensional feature set. The gains in efficiency are achieved by employing low-level color and texture features. Texture features are extracted using wavelet texture features, rather than the MSAR model in order to significantly reducing the feature dimensionality. High classification rates are achieved, despite the dimensional reduction, by using SVMs in a two-stage classification scheme. The first stage involved training color and texture SVMs based on image subblocks. The block-based classification rates will be lower than for the entire image. However, in the second stage, another SVM is used to train global color and texture features, respectively, for the entire image. Their method was trained and tested on a database of 1200 consumer photographs. Lazebnik *et al.* [118] proposed a method for recognizing scene categories based upon approximate global geometric correspondence. First, they computed so-called "weak features," which are oriented edge points (i.e., points whose gradient magnitude in a given direction exceeds a minimum threshold). The authors extracted edge points at two scales and eight orientations, for a total of $M = 16$ channels. These features are designed to obtain features which are similar to the gist or to a global SIFT descriptor of the image. For more discriminative features, they also utilized higher dimensional "strong features," which are SIFT descriptors of 16×16 pixel patches computed over a grid with a spacing of 8 pixels. Then, they performed k-means clustering of a random subset of patches from the training set to form a visual vocabulary. Finally, the resulting "spatial pyramid" is a simple and computationally efficient extension of an orderless bag of features of image representation. In their experiments, they confirmed that global representations can be surprisingly effective not only for identifying the overall scene but also for categorizing images as containing specific objects. Even when these objects are embedded in heavy clutter and vary significantly in appearance and pose. Zhou and Zhang [287] formalized Multi-Instance Multi-Label learning (MIML), where each training example is associated with not only multiple instances but also with multiple class labels. Then, they analyzed the relationship between multi-instance where an object has many alternative input descriptions (i.e., instances) and multi-label learning where an object has many alternative output descriptions (i.e., labels). Finally, they proposed the MIMLBOOST and MIMLSVM algorithms which achieved good performance in an application to scene classification.

An indoor scene recognition model is proposed by Quattoni and Torralba [187]. They presented an approach for indoor scene recognition which learned scene prototypes analogous to the constellation models. The goal of using image prototypes is to define a mapping between images and scene labels which can capture the fact that 1) images containing similar objects must have similar scene labels and 2) some objects are more important than others in defining a scene's identity. The authors also provided a large database that contains 15620 images organized into 67 indoor classes from an extensive range of fields. The images in the dataset have a minimum resolution of 200 pixels in the smallest axis and were collected from different sources: online photo sharing sites (Flickr), online image search tools (Google and Altavista), and the LabelMe dataset. Li *et al.* [129] introduced the concept of using an object as attributes for scene representation. They described complex real-world scenes by collecting their responses to a large number of object detectors, or object filters. These representations provided high-level semantic information rather than low-level image feature information. Object filters are introduced to characterize local image properties related to the presence/absence of objects. By using a large number of such object filters, the object bank representation of the image can provide rich information of the scene that captures much of the high-level meaning, which is more suitable for high-level visual recognition tasks. Li *et al.* [128] provided a high-level image representation, called the Object Bank (OB). OB is a representation of natural images based on objects, or more specifically, a collection of object sensing filters built on a generic collection of labeled objects. The authors explored how a simple linear hypothesis classifier, combined with a sparse-coding scheme, can leverage on this representation, despite its extreme high-dimensionality, to achieve superior predictive power over similar linear prediction models trained on conventional representations. This type of algorithms make their representation more efficient and scalable for large scene datasets and reveal semantically meaningful feature patterns. Pandey and Lazebnik [180] presented scene recognition with softly supervised object localization. They proposed an approach to describe the latent joint structure of scenes and objects using both the Latent SVM (LSVM) and Deformable Part-based Models (DPMs). They adapted DPM's for multi-class scene classification in a one-vs-all framework, where they trained a binary LSVM classifier for each class using images from all the other classes as negative data. At the testing stage, they labeled the test image with the class getting the highest response. Similar to [187], the root filter captured the holistic perceptual properties of the complete scene, while the region of interest corresponds to the most significant objects. The authors obtained representation with similar expressive power but much higher performance than that of Quattoni and Torralba [187] which requires a ground-truth Region Of Interest (ROI) annotations to get the best performance. While Pandey and Lazebnik [180] used the LSVM training process to determine the region of interest automatically. Bolovinou *et al.* [28] proposed bag of spatio-visual words which encode ordered spatial configurations of visual words in order to add context to the representation. In the first step, a feature space of visual words is built and each image is represented as a BoVW histogram by clustering of visual words' correlogram ensembles. Then, in the second step, a feature space of spatio-visual words is built by clustering of log-polar CoTrans descriptors, which encode ordered sets of visual words locally in an image. After applying the online spherical k-means algorithm on the set of CoTrans vectors the BoSVW model is produced where cluster centers constitute the novel spatio-visual words. Finally, the scene classification of an image is obtained by using the SVM classification algorithm on the concatenated BoVW and BoSVW image histogram representation. Juneja *et al.* [93] proposed a simple, efficient, and effective method to learn the part appearance and also to identify the part occurrences in images. They addressed this problem by learning parts incrementally, starting from a single part occurrence with an exemplary SVM. Then an initial model is refined by alternating mining for additional part instances and retraining. While this procedure requires training a sequence of detectors, the LDA technique is used to avoid mining for hard negative examples, eliminating the main bottleneck in detector learning, and enabling a very efficient part-learning algorithm. They also proposed entropy-rank curves to select distinctive parts among the ones that are generated by the part mining process. This criterion selected parts that are informative for a small proportion of classes. Contrastingly to other measures such as average precision, the resulting parts can then be shared by more than one object category. This is particularly important because parts should be regarded as mid-level primitives that do not necessarily have to respond to a single object class. Zuo *et al.* [299] presented an approach to learn transformation filter bank in order to transform pixel values of local image patches into features, which is called Discriminative and Shareable Feature Learning (DSFL). These learned filters are provided to: (1) encode common visual patterns of a flexible number of categories, and (2) encode discriminative and class-specific information. For this reason, DSFL approach learned a flexible number of shared filters in order to represent common patterns shared across different categories. To enhance the discriminative power, it forced the features from the same class to be locally similar, while features from different classes to be separable. Hu *et al.* [85] proposed an improved Unsupervised Feature Learning (UFL) algorithm based on spectral clustering, called UFL-SC. UFL-SC cannot only adaptively learn good local feature representations but also discover intrinsic structures of local image patches. Contrary to traditional UFL methods, UFL-SC mapped the original image patches embedded in the high-dimensional image space into a low-dimensional and intrinsic feature space by linear manifold analysis techniques. And then for feature encoding, UFL-SC learns a dictionary (e.g., using k-means clustering) on the patch manifold. The authors applied an explicit parameterized feature encoding method, i.e., triangle encoding with the learned dictionary on the same patch manifold in order to generate a feature representation for each local patch. Finally, the holistic feature representation of image scenes is obtained by building a BOW model of the encoded local features. Hayat *et al.* [74] suggested a new learnable feature descriptor named "spatial layout and scale invariant convolutional activations". It aims to deal with two important problems: scale variations and large-scale spatial layout deformations. For the scale invariance, the authors used a pyramidal image representation on which an image is resized to different scales, then the features are extracted across these scales. Whereas for the spatial layout invariance, they proposed a new feature description method based on modified CNN which consists of five convolutional layers and four fully-connected layers similar to AlexNet. The main difference resided on including the extra fully-connected layer, and that all of CNN modified layers are densely connected. Khan *et al.* [102] proposed a new method for indoor scene categorization that exploited rich mid-level convolutional features, unlike previous works that use the local or global spatial and appearance information. These mid-level representations are extracted from uniformly and densely image patches using Deep Convolutional Neural Networks (CNNs). Then, the extracted mid-level features are encoded in terms of their association with the codebooks of Scene Representative Patches (SRPs). Dixit *et al.* [48] suggested a new approach for scene classification based on semantic Fisher vectors (FVs). The authors obtained a Bag of Semantics (BoS) using trained CNN by extracting semantic descriptors (i.e., object class posterior probability vectors) from local image patches. The resulting representation is referred to as a semantic FV. The authors provided two implementations of a semantic FV. The first implementation modeled the BoS with a Dirichlet Mixture and computed the Fisher gradients for this model. Unfortunately, this approach is shown to be unsuccessful due to the difficulty of mixture modeling on a non-Euclidean probability simplex. The second implementation is derived using the interpretation of semantic descriptors as parameters of a multinomial distribution. A semantic FV is computed as a Gaussian Mixture FV in the space of the natural parameters. This representation outperformed other alternatives such as FVs of features from the intermediate CNN layers or a classifier obtained by fine-tuning the CNN. The FV represented an embedding for object classification probabilities. As an image representation, therefore, it is complementary to the features obtained from a scene classification CNN. Nogueira *et al.* [175] evaluated three possible strategies of exploiting CNN: (i) full-trained CNN, (ii) fine-tuned CNN, and (iii) pre-trained CNN used as feature extractors. In the first strategy (i), a CNN is trained from scratch obtaining specific visual features for the dataset. This strategy gave full control of the architecture and parameters, which tends to provide a more robust and efficient network. However, it required a large amount of data, since the convergence of the network is pruned to overfitting. In the second strategy (ii), pre-trained CNN performed fine-tuning of its parameters (i.e., filter weights) using the remote sensing data of interest. Usually, in this case, the earlier layers are preserved, as they encode low-level features, and final layers are adjusted to encode specific features of the data of interest. In the third strategy (iii), pre-trained CNN is used as a feature extractor, by removing the last classification layer and considering its previous layer (or layers) as a feature vector of the input data. Finally, the authors evaluated the performance of the three different strategies for exploiting the existing CNN and showed that fine-tuning tends to be the best strategy in different situations. Especially, using the features of the fine-tuned network with an external classifier, linear SVM in their case provided the best results.

6.2.3 3D recognition and categorization approaches

Recently, with the availability of consumer RGBD sensors such as the Microsoft Kinect, the use of depth information has been the subject of a number of important recent works for various vision-related tasks such as scene understanding and classification.

Silberman and Fergus [218] introduced a new RGBD dataset with densely labeled pairs of RGB and depth images which are captured using the Microsoft Kinect camera. This device is selected over other depth devices (e.g., time-of-flight and LIDARs cameras) due to its compactness, portability (i.e., after few modifications), accuracy, and its cheap price. Also, the Kinect camera used structured light methods to give an accurate depth map of the scene, which can be aligned temporally and spatially with the device's webcam. All these advantages make use of this device viable in numerous vision applications, such as assisting the robot navigation. Furthermore, the authors provided on this dataset scene classification using the spatial pyramid matching scheme as well as the multi-class segmentation task using several features. Gupta *et al.* [68] were interested in contour detection, bottom-up grouping and semantic segmentation using both RGB and depth information for scene classification. The authors formulated the contour detection as a binary pixel classification that separates contour from non-contour pixels. Then, they learned classifiers for each orientation channel. They also proposed a generic technique for long-range amodal completion of surfaces as well as the hierarchical segmentation by merging regions of the initial over-segmentation. Finally, the scene classification is based on the semantic segmentation maps, spatial pyramid, and SVM. Shotton *et al.* [216] proposed a new approach for camera relocalization in RGBD images. The approach is based on a regression forest that predicted the correspondences from any image pixel to 3D points in the scene's world coordinate frame. Scene recognition is also introduced using the pose optimization algorithm, separately for each scene's coordinate regression forest (SCoRe Forest). Using RANSAC optimization, the scene with the largest number of inliers (i.e., the lowest energy) is selected as the result. Moreover, the authors provided a new RGBD dataset, named "7 Scenes", to evaluate their technique and compare it with other approaches. All scenes were acquired from a handheld Kinect RGBD camera at 640×480 resolution. Then, an implementation of the Kinect Fusion system is used in order to obtain the "ground truth" camera tracks. Tao *et al.* [237] suggested a Rank Preserving Sparse Learning (RPSL) approach for scene classification task. RPSL takes into consideration four aspects: the first aspect maintained the rank order information of the within-class samples in a local patch while it ignored the rank order information of the between-class samples; the second aspect maximized the margin for the between-class samples on a local patch, the third aspect introduced the L1-norm penalty to obtain the sparse representation, and the final aspect modeled the classification error minimization that used the least squares error minimization. The authors compared RPSL to the classical dimension reduction algorithms, such as PCA, linear discriminant analysis, discriminative locality alignment, supervised locality preserving projections, and sparse PCA, and showed that RPSL presented many competitive as well as attractive properties for Kinect-based scene classification. Feng *et al.* [54] represented a new manifold-learning-based discriminative feature learning method for scene classification called Discriminative Locality Alignment Network (DLANet). DLANet adopted the PCANet structure which learned the convolutional filter bank through PCA in order to learn the local features. The authors argued that DLA can cope with the non-linearity of the distribution of samples while preserving the discriminative information and enhance the importance of marginal samples for discriminative subspace selection. DLANet trained the filters in CNN with the manifold assumption and provided more effective discriminative information. Finally, the classification task is performed by combining linear SVM and Locality-constrained Linear Coding-Spatial Pyramid Matching (LLC-SPM). Finally, the authors showed that DLANet features can generate a more efficient image representation with coding and spatial pooling. Lu *et al.* [142] proposed semi-supervised multitask learning for scene recognition. The approach modeled a multitask learning in order to integrate different resolutions of the scene images. For the differences of different resolutions, the local features of different resolutions are mapped to a common subspace by using multitask technique, since it can improve the performance of multiple related tasks by exploiting the intrinsic relationships among multiresolution. In this case, the differences between local features can be reduced and the mapped feature of samples from different resolutions can be learned, which can preserve the structural commonness of different resolution images. Also, the approach built a model of Sparse Feature Selection-based Manifold Regularization (SFSMR) with the aim to select the optimal information and maintain the underlying manifold structure of data. Finally, multi-class classification is achieved with SVM classifier using the one-versus-all method. The approach presented three main advantages. First, it took multiresolution images as multiple-related tasks and used the common knowledge of multiple tasks. Second, the underlying manifold structure of each feature data is preserved and the optimal features can be chosen. Third, by using the $L_{2,1}$ norm term and the trace norm term, the correlation of different features at multiresolution can be exploited and the information from different tasks can be transferred among multiple tasks. Wang *et al.* [261] investigated a framework that allows greater spatial flexibility, in which the Fisher Vector (FV) encoded distribution of local CNN features. These features are extracted from an augmented pixel-wise representation comprising multiple modalities of RGB, HHA (i.e., Horizontal disparity, Height above ground, and Angle between the local surface normal and direction of inferred gravity), and surface normals to capture more information of the geometry. Also, the authors make two important postulates: (1) component sparsity, and (2) modal non-sparsity. The component sparsity showed that only a small variety of region proposals and their corresponding FV Gaussian Mixture Model (FV GMM) components contributed to scene discriminability. The modal non-sparsity showed that for these key discriminative components, all modalities will significantly contribute to the discriminability because they provided important complementary information. By combining the regression results of their approach based multi-modal FV features and the full-image based multimodal CNN features, the authors achieved state-of-the-art scene classification performance on the SUNRGBD dataset and the NYU Depth Dataset V2. Liao *et al.* [132] addressed the problem of scene classification by using deep neural networks to incorporate object-level information. They developed a scene classification model with regularization of Semantic Segmentation based on the well-known CNN architecture Alexnet, called SS-CNN. Finally, they applied the SUN RGBD dataset trained model to a mobile robot, which captured images to classify scenes in real-world environment demonstrating the generalization ability of SS-CNN algorithm. Song *et al.* [228] proposed depth CNNs for RGBD scene recognition task. The authors focused on the bottom layers and proposed an alternative strategy to learn depth features combining local weakly supervised training from patches followed by global fine-tuning with images. The aim of this strategy is learning very discriminative depth specific features with limited depth images, without resorting to Places-CNN. Moreover, they proposed a modified CNN architecture to further match the complexity of the model and the amount of data available. For RGBD scene recognition, pre-trained RGB and depth networks are combined into a new network and fine-tuned with RGBD image pairs. Finally, they showed experimentally that their framework achieved state-of-the-art accuracy on NYU 2 and SUN RGBD in both depth only and combined RGBD data. Cai and Shao [34] proposed a new CNN-based Local Multi-modal feature learning framework (LM-CNN) for RGBD scene classification. LM-CNN can effectively capture much of the local structure from the RGBD scenes instead of simply concatenate RGB and depth features without exploring the correlation and complementarity between raw RGB and depth images. Also, LM-CNN can automatically learn a fusion strategy for the object-level recognition step instead of simply training a classifier on top of features extracted from both modalities.

6.3 Conclusion

In this chapter, we presented a very brief overview of the human visual attention mechanisms as well as the state-of-the-art scene classification approaches. The relevant points to the remainder of the work presented here are the distinction between bottom-up and top-down visual attention. Also, humans demonstrated the ability at instantly capturing the gist of a scene and report its category (i.e., indoor or outdoor) for just a fraction of a second. The gist has recently been the subject of comprehensive research efforts and is used in many scene classification approaches. In the next chapter, we will present our contribution which is based upon gist representation for indoor scene classification.

Chapter 7

Discriminative Deep Belief Network for Indoor Environment Classification using Global Visual Features

"Every frame and every scene has to have an intention."

Mira Nair

Contents

INDOOR environment classification, also known as indoor environment recognition, is a highly appreciated perceptual ability in mobile robots. In this chapter, we present our approach [295] which is centered on biologically inspired methods for representation and classification of indoor environments. First, global visual features are extracted by using the GIST descriptor, and then we use the subsequent features for training the Discriminative Deep Belief Network (DDBN) classifier. DDBN employs a new deep architecture which is based on Restricted Boltzmann Machines (RBMs) and the joint density model. The back-propagation technique is used over the entire classifier to fine-tune the weights for an optimum classification. Our major contributions are as follows:

- we develop an indoor environment classification system by using biologically inspired methods based on GIST features and deep architectures;
- we extract global visual features for both real and synthetic datasets using the GIST descriptor;
- we introduce the Restricted Boltzmann Machines (RBMs) and the joint distribution that constitute the Discriminative Restricted Boltzmann Machines (DRBMs);
- we use different sampling methods in DRBMs: Contrastive Divergence (CD), Persistent Contrastive Divergence (PCD), and Free Energy in Persistent Contrastive Divergence (FEPCD);
- we introduce DBN and its discriminative ability using backpropagation strategy to optimize the classification results.

7.1 Introduction

A new lifestyle, including Human-Robot Interaction (HRI) in various environments such as in homes and offices, is anticipated in the future. For robots to become useful for humans, it is obligatory that they achieve the capability to develop a prompt understanding of the neighboring environment. Such ability is vital for tasks which involve autonomous movement in various conditions as well as in constantly altering situations [290]. The main question that arises in this context is how to represent and classify environments to be understood by robots?

Environment representation has been broadly considered in areas of cognitive psychology and computer vision. In cognitive psychology, several studies show that humans are capable of recognizing complex visual scenes within 1/20 of a second [23], independent of the scale of objects in the scene. This property of a human visual system is called "gist". Such notions are called global or holistic representation that stimulates the top-down knowledge and guides the visual analysis for advanced scene interpretation. In this representation, the perceptual information about objects and their locations is overlooked. It models the holistic representation that provides a description of the scene which includes the probable semantic category.

On the other hand, there are some bottom-up models which are inspired by the cognitive concepts [1, 87, 120, 147]. Itti *et al.* [87] presented a visual attention model which is based on the neuronal architecture and behavior of the primate visual system. This model used three feature channels: color, intensity, and orientation. The input image is sub-sampled into a Gaussian pyramid and then each pyramid level is decomposed into channels for red, blue, green, yellow, intensity, and local orientations. The center-surround "feature maps" of these channels are constructed and normalized. Finally, conspicuity maps are linearly combined to constitute the saliency map. Le Meur *et al.* [120] introduced a coherent computational model of the bottom-up visual attention. Their approach is mainly based on contrast sensitivity functions, visual masking, perceptual decomposition, and center-surround interactions. Marat *et al.* [147] proposed a new spatiotemporal saliency model to predict eye movement in videos. This model extracted two signals from the video that corresponds to parvocellular and magnocellular cells of the retina. Thereafter, these signals are split by cortical filters into elementary feature maps which are used to form static and dynamic saliency maps. Finally, these maps are fused into a spatiotemporal saliency map. Aboudib *et al.* [1] suggested a new framework for visual information acquisition and representation. Their model provided a method that takes the distance between an image and the viewer. Then, it reduced the amount of visual information acquired using a new scheme for emulating retinal sampling and the cortical magnification effects which are observed in the ventral stream.

Specifically, the ability to identify or recognize indoor environment such as office, bedroom, and so on, is highly valuable for performing imperative tasks in mobile robotics. Artificial Neural Networks (ANNs) are used for several classification applications [35, 89, 191]. They are theoretical and computational models, inspired by the configuration of the human brain. There are varieties of network architectures and learning approaches that can be pooled to develop neural networks with diverse computational capabilities. However, until 2006, researchers didn't know how to train neural networks to surpass more traditional approaches, except for a few specialized applications such as speech recognition and natural language processing. In 2006, the procedures for learning in neural networks were suggested, and these techniques are now recognized as *deep learning*.

Deng and Yu [46] defined deep learning as a class of machine learning systems that process information for unsupervised or supervised feature extraction using numerous layers of non-linear processing. Those features are later employed for pattern analysis and classification. Deep architectures attempt to extract features in various stages of abstraction for enabling a system to pick up complex functions that map the input data to the output directly. The stages in these statistical models consist of discrete levels of concepts where lower-level concepts define the higher-level ones [20].

Serre *et al.* [214] showed the evidence that the brain of a mammal is organized as a deep architecture. A specified input is characterized by various levels of abstraction where every level relates to a diverse area of cortex. Researchers used the deep architecture concept in neural networks for training new deep MLPs which are stimulated by the biological depth of the brain. Such deep models involved numerous layers and parameters that require being learned through the complex process. To deal with this problem, Hinton *et al.* [81] suggested a Deep Belief Network (DBN) with multiple layers of hidden units. DBN is a graphical model comprising undirected networks at the top hidden layers and directed networks in the lower layers. The learning algorithm uses greedy layer-wise training by stacking restricted Boltzmann machines (RBMs). It comprises a hidden layer for modeling the probability distribution of perceptible variables. This model is used for several classification tasks such as object classification [179], speech recognition, and phone recognition [169].

In this chapter, we propose a new classification approach for indoor environments based on bio-inspired methods. First, we extract the global visual features from GIST descriptor that operates like the human visual system by shortly extracting the essential information in the image regardless of its complexity. Then, we learn these resulting features using DDBN which is stimulated by the biological depth of brain and provides discriminative power for pattern classification. Moreover, the objective of our approach is to ensure a classification system that performs as CNN in term of recognition rate but requires a short computing time. The performances of the combination of GIST descriptor and DDBN in term of computational complexity remains better than CNN since the CNN used many layers (i.e., convolution, max-pooling, and fully-connected) to extract then classify the features.

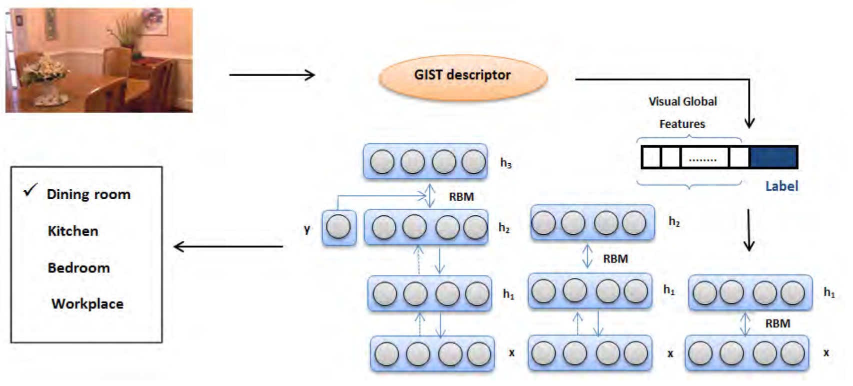


Figure 7.1: Our approach for indoor environment classification using global visual features and DDBN.

The general approach is achieved in Figure 7.1 as follows:

1. extracting global visual features for all training set using GIST descriptor;
2. input layer x takes the extracted global visual features;
3. training RBM using CD, PCD or FEPCD sampling:
   - training the first RBM;
   - training the second RBM using the training data resulting from the first RBM learning;
   - training a joint density model through discriminative RBM and then each visible label is tested with a test vector. The label which contains the least energy is selected as the best corresponding class.
4. using the backpropagation technique over the entire classifier to optimize the weights.

7.2 Scene representation: GIST descriptor

Scene representation is an important task for many of our most valued behaviors such as classification, navigation, localization, and reasoning with the world around us. So, what is a "visual scene"? And how to identify it?

7.2.1 What is a visual scene?

Aude Oliva [176] defined a visual scene as a view in which objects and surfaces are arranged in a meaningful way, for example, a kitchen, a street, or a forest. Scenes contain elements arranged in a spatial layout and can be viewed at a variety of spatial scales (e.g., the up-close view of an office desk or the view of the entire office).

7.2.2 What is the gist of the scene?

Considering an observer that watches television and flips rapidly through the channels, with a mere glimpse of each image, the observer can grasp each one's meaning (e.g., a footballer, a serial, the news, cartoons) independently of the clutter and the diversity of details. With just a glance at a complex real-world scene, an observer can comprehend a variety of semantic and perceptual information. This refers to the gist of a scene [60].

7.2.3 Global descriptors

Global descriptors use all pixels of an image to compute a signature. Therefore, the vector obtained is sensitive to the objects which are present in the scene, to their positions, to the global illumination, and so on. Although using all the pixels of an image, it is possible to include in these descriptors the geometric information by splitting the image into sub-images. Then the overall description can be obtained by concatenating the description of each sub-image. This is called *coarse geometry*. As shown in Figure 7.2, the image is split according to a regular grid (i.e., in general, each block is less than 100 pixels) to incorporate geometric information into a global description. The advantage of regular segmentation of the image is to propose a final representation, for example, a vector of local signatures, which indirectly encodes information on the spatial organization of the scene. Global descriptors capture the coarse version of the principal contours and textures of the image that is still detailed enough to recognize the image's gist. One of the main advantages of the global descriptors lies in computational efficiency, there is no need to group the component of the image in order to represent the spatial configuration of the scene.

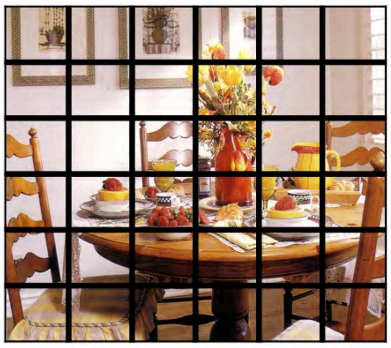


Figure 7.2: Regular grid.

7.2.4 GIST descriptor

Originally presented by Oliva and Torralba [178], the GIST descriptor stems out of a series of psychological and computer-based studies about the classification of scenes. The human visual system can classify an image in a short time regardless of its complexity. The goal was to create a system which could replicate the same operation in artificial machines. This process can be achieved through the process extracting the essential information in the image (i.e., the gist).

The GIST descriptor is based on the extraction of the spatial envelope of the image. The spatial envelope represents a set of general scene properties that can be used for understanding the semantic class of the scene without the necessity to recognize the objects in it. Oliva and Torralba [177] introduced in their work the following five spatial envelope properties.

- Degree of naturalness: the structure of a scene strongly differs between artificial and natural environments. Straight horizontal and vertical lines dominate artificial structures while most natural landscapes have textured zones and undulating contours. However, scenes which have a distribution of edges commonly found in natural landscapes would have a high degree of naturalness.

- Degree of openness: a scene can have a closed spatial envelope full of visual references (e.g., a forest, a mountain), or it can be vast and open to infinity (e.g., a coast, a highway). The existence of a horizon line and the lack of visual references confer to the scene a high degree of openness.

- Degree of roughness: it depends on the size of elements at each spatial scale, their abilities to build complex elements and their relations between elements that are also assembled to build other structures, and so on.

- Degree of expansion: artificial structures are mainly composed of horizontal and vertical structures. Therefore, according to the observer's point of view, structures can be seen from different perspectives. The convergence of parallel lines gives the perception of the depth gradient of the space. A flat view of a building would have a low degree of expansion.

- Degree of ruggedness: it refers to the deviation of the ground with respect to the horizon (e.g., from open environments with a flat horizontal ground level to mountainous landscapes with a rugged ground). A rugged environment produces oblique contours in the picture and hides the horizon line. Most of the artificial environments are built on flat ground. Therefore, rugged environments are mostly natural.

The GIST features are computed by converting the input image to grayscale, normalizing the intensities and locally scaling the contrast. The resulting image is then spread over a grid ($M \times M$ cells) on several scales. The response of each cell is computed using a series of Gabor filters. All the cell responses are combined to form the feature vector, which describes the image in its fully fledged form. The GIST descriptors present the optimum results using the default setup consisting of a filter bank at 4 scales and in 8 orientations. These descriptors, being $4 \times 8 \times 16 = 512$ dimensional, are computed for the grayscale variant. Figure 7.3 shows the GIST descriptor for indoor environment scenes.

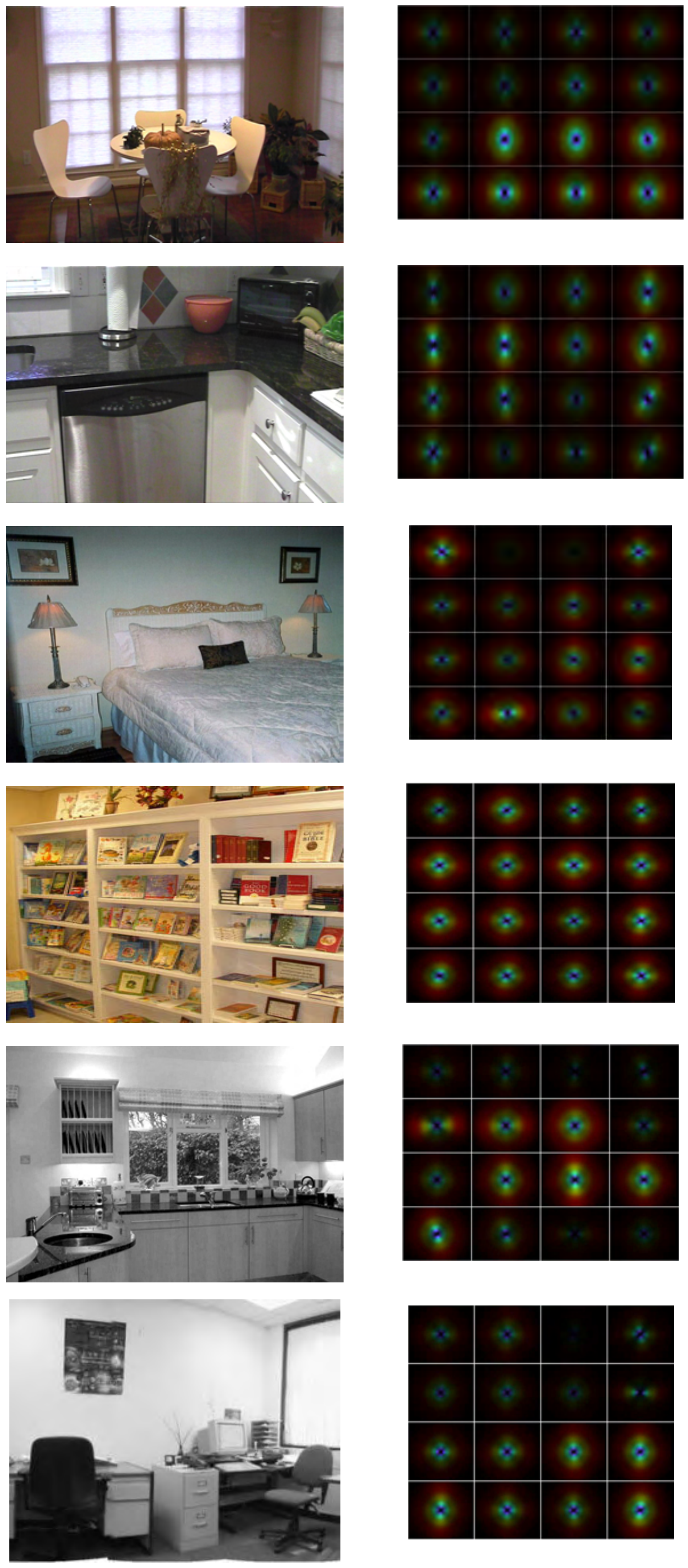


Figure 7.3: The example of GIST descriptors for indoor environment scenes.

7.3 Scene classification using DDBN

In this chapter, we performed scene classification using Discriminative Deep Belief Network (DDBN). DDBN aims at letting every RBM model in the structure to obtain a diverse representation of data. In other words, after RBM is trained, the activity values from the hidden units act as the training data for a higher-level RBM learning. In DDBN, we need to use a DRBM in the last layer as a classifier for obtaining labels from the input data. The input layer has a N number of units which is equivalent to the quantity of sample data x. The label layer has C representing y as the number of classes. DDBN trains a joint density model through discriminative RBM and then each visible label is tested with a test vector. The label which contains the least energy is selected as the best corresponding class. Afterward, we use the backpropagation technique through the entire classifier for fine-tuning the weights for optimal classification. Also, we used Gaussian-Bernoulli Restricted Boltzmann Machines (GBRBMs) (Section 2.2.2) which modeled real-valued inputs that are very appropriate to our GIST descriptors as input data. Finally, Contrastive Divergence (CD), Persistent Contrastive Divergence (PCD), and Free Energy in Persistent Contrastive Divergence (FEPCD) are used to train the GBRBMs (Section 2.2.2).

7.4 Experimental results

7.4.1 Datasets

We test the performance of our indoor environment classification system on Visual Place Categorization (VPC), MIT, and 15 Scenes datasets.

Visual Place Categorization (VPC)

Visual Place Categorization (VPC) dataset[1] consists of videos captured autonomously using a rolling tripod plus a HD camcorder (JVC GR-HD1) to mimic a robot. VPC collects videos from six home environments. In our work, we are interested in videos that provide 360° views of rooms. In this experiment, the camcorder was fixed inside each room to take video by slowly rotating it on the tripod. Each frame from the videos has a JPEG format (95% quality and 1280×720 in resolution). Figure 7.4 depicts the sample images of VPC dataset.

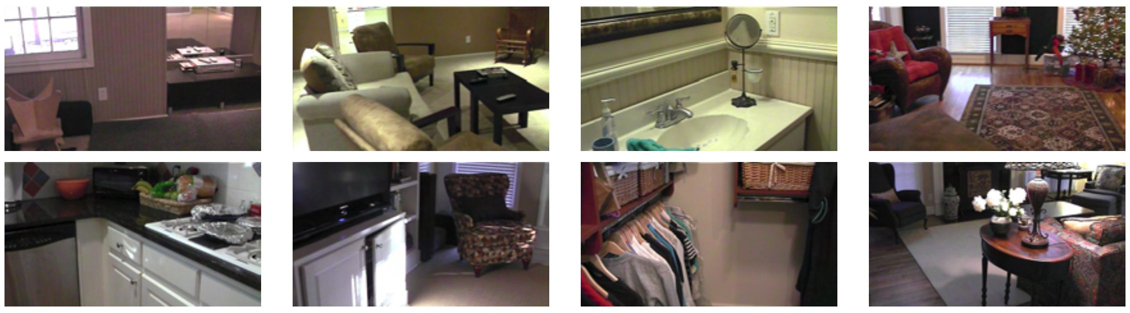


Figure 7.4: Selected images from VPC dataset that contains sevral categories.

[1]http://categorizingplaces.com/dataset.html

MIT dataset

MIT dataset [187] is a large dataset of indoor scene categories. These images are collected from different sources such as LabelMe dataset, online image search engines, and photos. It consists of 15 620 images which are organized into 67 categories. The images which are shown in Figure 7.5 have a minimum resolution of 200 pixels in the minimum axis.

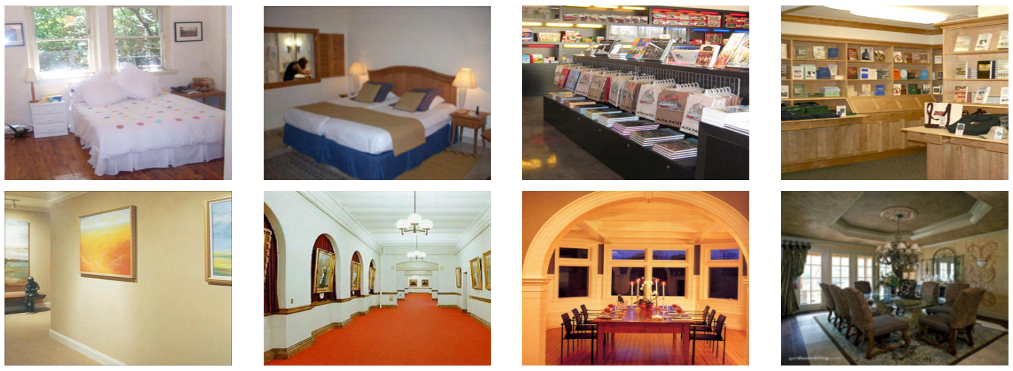


Figure 7.5: Selected images from MIT dataset.

15 Scenes dataset

15 Scenes dataset [118] contains fifteen scene categories in which each one is composed of 200 to 400 images. Each image size is about 300 × 250 pixels. The dataset includes COREL image collection, Google, and personal photographs. 15 Scenes dataset contains both indoor and outdoor scenes. As depicted in Figure 7.6, we used only indoor environment (e.g., kitchen, living room, office, store, and bedroom) which are specific for our application.

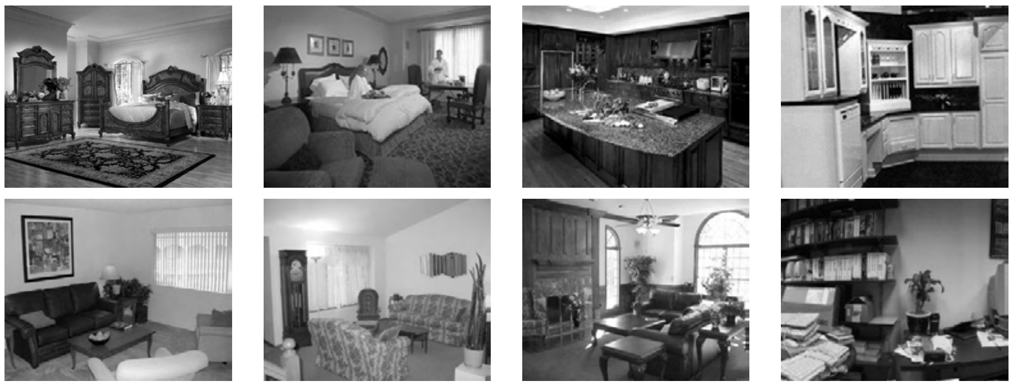


Figure 7.6: Selected images from 15 Scenes dataset that contains sevral categories.

7.4.2 DDBN experimental setup

We used a DDBN with three hidden layers, each containing different hidden units to define 512-200-200-1000, 512-300-300-1500, and 512-500-500-2000 structures. The weights of each layer are distinctly trained using a fixed number of epochs. This approach trains RBMs one after another and uses their training data resulting from the training stage in the next RBM using CD, PCD or FEPCD sampling methods. The last layer trains a joint density model with a discriminative RBM. To optimize the classification, we use the backpropagation technique over the whole classifier to fine-tune the weights. Table 7.1 illustrates the DDBN characteristics which are used in our experiments.

Table 7.1: DDBN characteristics that are used in our experiments.

Characteristics	**Values**
Input layer units	size of GIST descriptor (i.e, 512 features)
Hidden layers	3
Learning rate	0.01
Epochs	200
Sampling methods	CD, PCD, and FEPCD

7.4.3 Scene classification results

Figure 7.7 shows the best training performance on VPC dataset. It indicates the iterations at which the validation performance reaches a minimum mean squared normalized error (MSE) performance criterion (Section 5.2.3 in Chapter 5 provided the MSE definition). In all the VPC experiments, the best performance is obtained with PCD and FEPCD sampling.

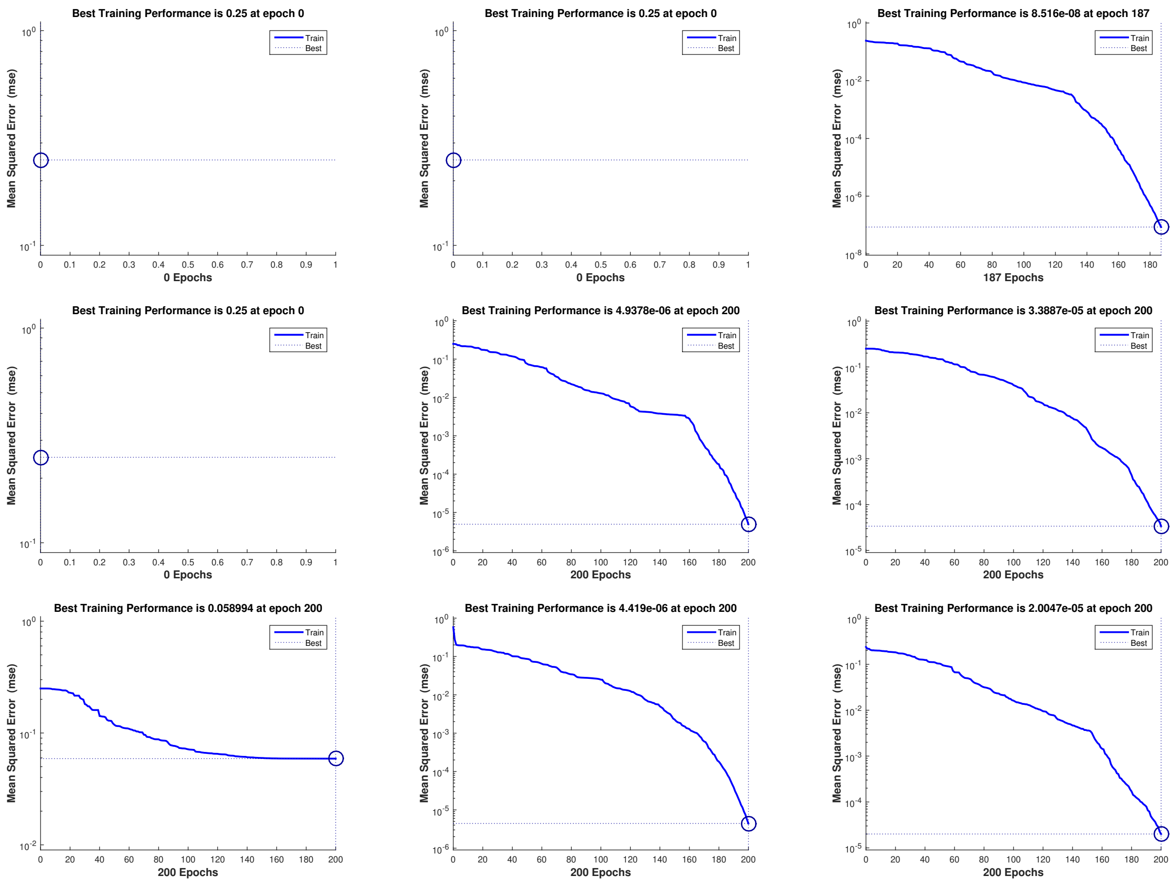


Figure 7.7: Best training performance on VPC dataset. Each row from top to down represents 512-200-200-1000, 512-300-300-1500 and 512-500-500-1500 respectively. Each column represents CD, PCD and FEPCD sampling methods respectively.

Also, Table 7.2 illustrates the classification errors Before and After using a Back-Propagation technique (BBP/ABP). We notice that performance errors ABP and BBP techniques in

Sampling methods	CD			PCD			FEPCD		
Values	BBP	ABP	ACC	BBP	ABP	ACC	BBP	ABP	ACC
512-200-200-1000	0,536	0,536	46.31%	0,739	0,739	26.05%	0,842	0	100%
512-300-300-1500	0,513	0,513	48.68%	0,589	0	100%	0,715	**0**	**100%**
512-500-500-2000	0,510	0,236	76.31%	0,713	0	100%	0,7	0	100%

Table 7.2: Classification errors and accuracies on VPC dataset for different DDBN structures.

DDBN-CD and DDBN-PCD with 512-200 200-1000, and DDBN-CD with 512-300 300-1500, are similar. This is due to the small number of hidden units that were equal to 200 or 300 in the second and the third layers. In contrast, when we increase the number of hidden units, the error ABP decreases. Table 7.2 shows also that our indoor environment classification approach works perfectly with the accuracy rate of 100% in the case of DDBN-FEPCD. This result is efficient due to VPC dataset which contains 360° viewpoints of rooms.

Figure 7.8 shows that FEPCD sampling method remains the best one and provides a minimum value of MSE in all DDBN architectures. Table 7.3 illustrates the classification errors BBP and ABP techniques on MIT dataset. It is observed that the error decreases after using the backpropagation technique, especially with 512-300-300-1500 DDBN-FEPCD structure.

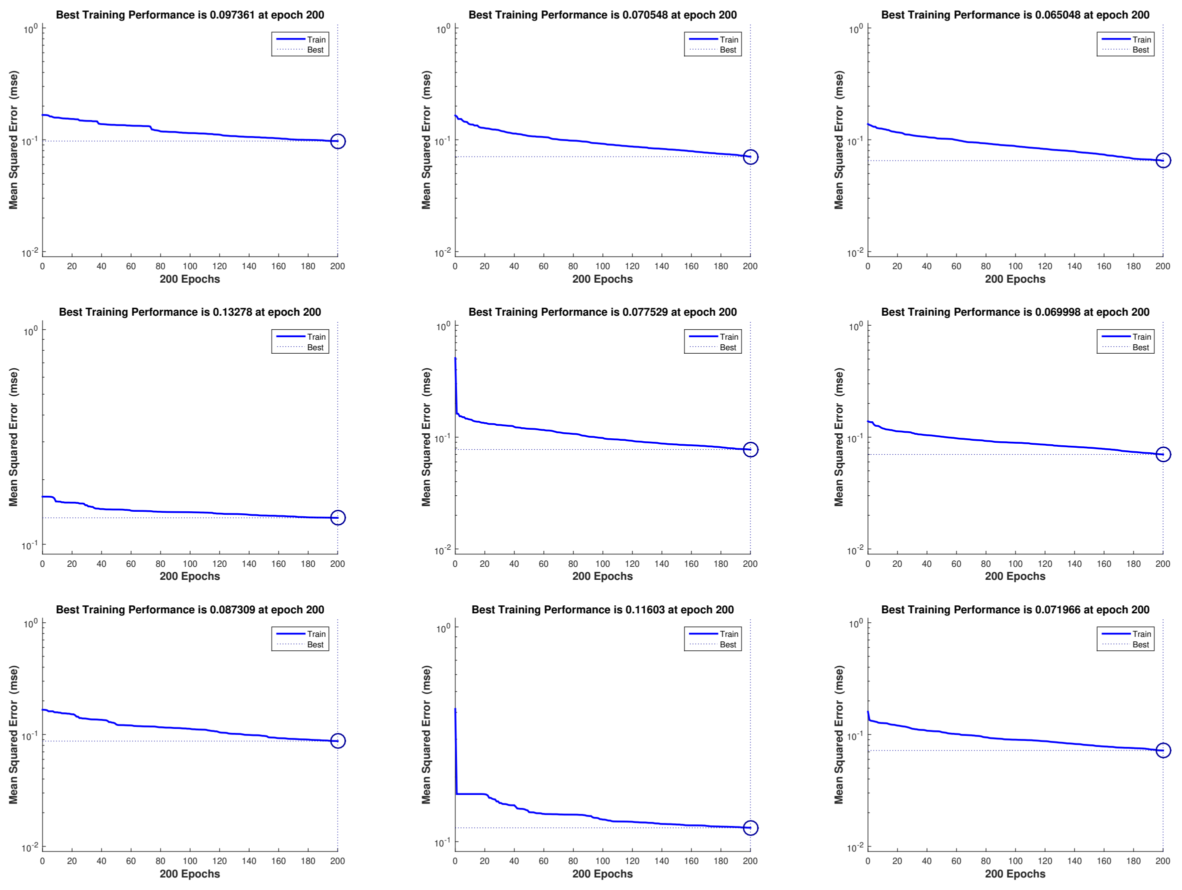


Figure 7.8: Best training performance on MIT dataset. Each row from top to down represents 512-200-200-1000, 512-300-300-1500 and 512-500-500-1500 respectively. Each column represents CD, PCD and FEPCD sampling methods respectively.

Table 7.4 provides the comparison between our approach and DPM-GC-SP [180] approach. Pandey and Lazebnik [180] presented scene recognition with softly supervised ob-

Sampling methods	**CD**			**PCD**			**FEPCD**		
Values	BBP	ABP	ACC	BBP	ABP	ACC	BBP	ABP	ACC
512-200-200-1000	0,687	0,484	51.54%	0,649	0,374	62.53%	0,674	0,380	61.97%
512-300-300-1500	0,681	0,645	35.49%	0,743	0,380	61.97%	0,667	**0,366**	**63.38%**
512-500-500-2000	0,670	0,402	59.71%	0,732	0,5648	43.52%	0,688	0,380	61.97%

Table 7.3: Classification errors and accuracies on MIT dataset for different DDBN structures.

ject localization. They proposed an approach to describe the latent joint structure of scenes and objects using both the Latent SVM (LSVM) and Deformable Part-based Models (DPMs). They adapted DPM's for multi-class scene classification in a one-vs-all framework, where they trained a binary LSVM classifier for each class using images from all the other classes as negative data. At the testing stage, they labeled the test image with the class getting the highest response. In our work, we used both GIST descriptors and DDBN classifier to perform scene classification. As shown in Tables 7.3 and 7.4, we compared the 512-300-300-1500 DDBN structure with DPM-GC-SP [180] approach. The above approaches are tested on MIT dataset. We extract six categories from MIT dataset to validate our approach. Our method can outperform the DPM-GC-SP [180], the results 63.38% are significant and consistent compared to 40% from [180] approach (56% dining room, 67% corridor, 52% kitchen, 20% living room, 10% bedroom and 35% bookstore).

DPM-GC-SP [180]	**DDBN-CD**	**DDBN-PCD**	**DDBN-FEPCD**
40 %	35.49%	61.97%	**63.38%**

Table 7.4: Comparison results between our approach with 512-300-300-1500 DDBN structure and DPM-GC-SP approach [180].

As depicted in Figure 7.9, the minimum MSE 0.038079 is achieved with FEPCD sampling method and especially with 512-300-300-1500 DDBN architecture at epoch 200. In general, the use of PCD sampling performs better than CD sampling, and FEPCD outperforms PCD sampling in all DDBN structures. This result is obvious since FEPCD uses free energy as the criterion for goodness of a chain to obtain the best samples from the generative model that are able to compute the gradient of the training data's log probability. Table 7.5 shows that our indoor environment classification approach works perfectly on 15 Scenes dataset with the accuracy rate of 69% in the case of DDBN-FEPCD.

Sampling methods	**CD**			**PCD**			**FEPCD**		
Values	BBP	ABP	ACC	BBP	ABP	ACC	BBP	ABP	ACC
512-200-200-1000	0,694	0,514	48.52%	0,742	0,603	39.67%	0,798	0,372	62.73%
512-300-300-1500	0,640	0,627	37.26%	0,635	0,311	68.90%	0,624	**0,305**	**69.43%**
512-500-500-2000	0,608	0,313	68.63%	0,745	0,364	63.53%	0,691	0,380	61.93%

Table 7.5: Classification errors and accuracies (ACC) on 15 Scenes dataset for different DDBN structures.

In summary, the experiments on VPC dataset show that our approach of indoor environment classification works perfectly on a real-world dataset. This result is evident since the VPC dataset consists of a large set of images that are captured at different viewpoints, and objects are described well enough in images. Therefore, with better training data, the results obtained are significantly high with this dataset. These results assure that our proposed approach, which is based on GIST descriptor for global visual extraction and DDBN for feature classification, performs on indoor environment classification using 512-300-300-1500 DDBN structure and FEPCD sampling. FEPCD outperforms PCD and CD in terms of accuracy although its computational complexity is high and takes a relatively long time in training as compared to the other two methods. Our next goal will be to optimize the performance of FEPCD in order to reduce the computational complexity.

In chapter 10, we proposed a method of exploring indoor environments by an autonomous mobile robot, as well as building topological maps based on global visual attributes. This method takes advantage of the small size of the GIST descriptors, and the ease of their calculation. We also make use of omnidirectional images to build a single global visual descriptor showing an entire room and. However, our indoor environment classification can be applied for mapping and localizing a mobile robot in its environment since a robot is able to recognize its current place.

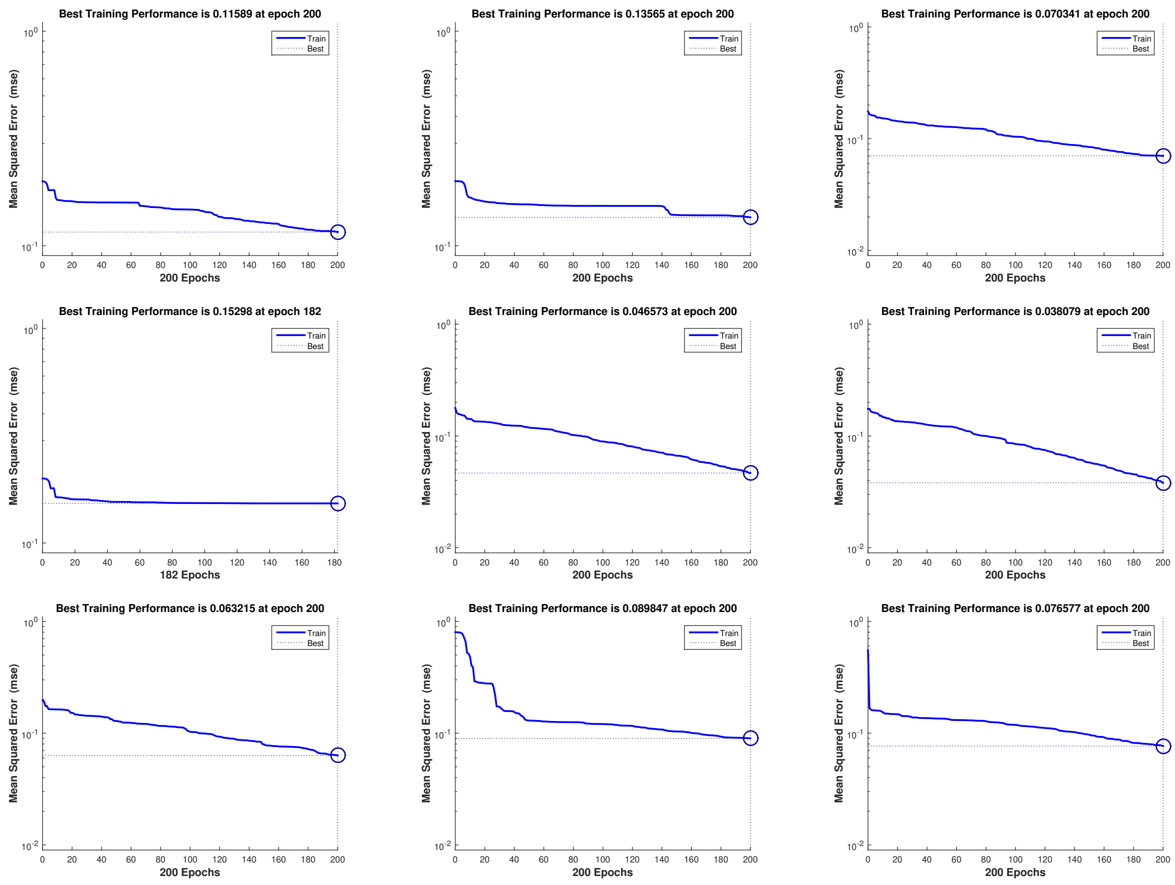


Figure 7.9: Best training performance on 15 Scenes dataset. Each row from top to down represents 512-200-200-1000, 512-300-300-1500 and 512-500-500-1500 respectively. Each column represents CD, PCD and FEPCD sampling methods respectively.

7.4.4 Comparison to CNN architecture

Our approach trains RBMs one after another and uses their training data resulting from training stage in the next RBM using CD, PCD or FEPCD training methods. The last layer trains a joint density model with a discriminative RBM. We use the backpropagation technique through the whole classifier to fine-tune the weights to optimize the classification result.

CNN experimental setup

All experiments are performed using Keras[2] python library which is running on top of either Theano or TensorFlow libraries. Keras provides an easy and fast prototyping, both CPU and GPU implementation, and some deep learning algorithms such as convolutional networks and recurrent networks. We have worked with the same subsets of VPC, MIT, and 15 Scenes datasets used with DDBN approach in the previous experiments. Each dataset is split into the train, validation, and test folders of images and trained using a Xeon(R) 3.50 GHz CPU 32 GB RAM and K2000 Nvidia card on Ubuntu 14.04 operating system. We use only the CPU device because of the limited graphic memory of our GPU card which is crucial for training CNN architecture. In the image pre-processing step, we reshape all the datasets into size 224×224 to be compliant to the standard input of CNN. Moreover, the weights of the architecture are distinctly trained using a fixed number of epochs equal to 200. Table 7.6 summarizes the general characteristics used in our CNN experiments.

Characteristics	Values
Optimizer	Stochastic Gradient Descent (SGD)
Learning rate	0.01
Learning rate decay	1e-6
Momentum	0.9
Loss	categorical cross entropy
Batch size	32
Dropout	0.5
Epochs	200

Table 7.6: Keras characteristics used in our CNN comparison.

Pre-trained CNN

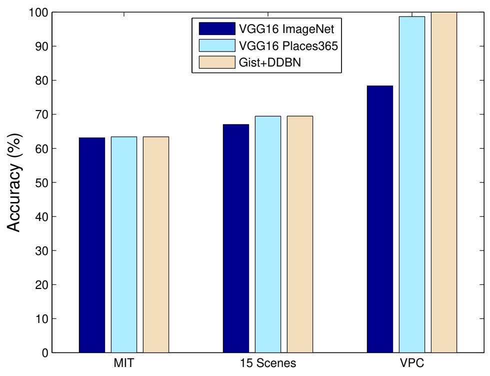
Many pre-trained models which are usually trained on the large computer vision datasets [44, 284, 285] can be used for computer vision and robotic applications such as object recognition [50], scene categorization, object detection, and segmentation [70]. ImageNet [44] dataset contains several classes such as animal, furniture, flower, food, and a vehicle which their features are already learned by the CNN model. However, the pre-trained model VGG16-ImageNet is not used only for this range of features, but it can work perfectly in problems featuring classes which are absent from ImageNet. Places [284] represents the largest scene-centric image dataset which collects 10 million scene photographs, labeled with 476 scene semantic attributes and categories from three macro-categories: indoor, nature, and urban environments. The Places365 benchmark was trained on only 365 categories of Places dataset with more than 4000 images. After, initializing the convolution blocks of the VGG16 model, we fine-tune the weights of the pre-trained network by continuing the backpropagation. We only fine-tune some high-level layers of the network. This is motivated by the observation that the last layers of the CNN are progressively more specific to the details of

[2]https://keras.io/

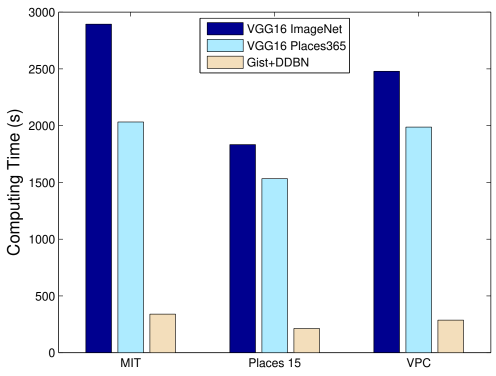
the classes contained in the original dataset. The details of fine-tuning CNN are provided in Chapter 8 (Section 8.3.3) which presents an RGBD scene classification approach based on pre-trained CNN.

DDBN vs. CNN

In this subsection, we provide the classification and the computational performance of our approach that uses both GIST descriptor and DDBN. We also compare the obtained results with the fine-tuned CNN which is initialized from pre-trained network weights (VGG16 pre-trained).



(a)



(b)

Figure 7.10: The comparison between CNN and DDBN on MIT, 15 Scenes, and VPC datasets. (a) shows the accuracy and (b) represents the computing time (CPU device) of CNN instead DDBN.

As shown in Figure 7.10, the accuracy obtained from our approach slightly outperforms the obtained accuracy as compared to VGG16-ImageNet and VGG16-Places365. However, the significant difference is the computing time, as CNN requires adequate time for training data, because of this deep architecture, which contains 16 layers instead of DDBN which used only three hidden layers. In this current work, we did not perform the experiments on GPU, but we believe that the proposed structure using DDBN will still be computationally efficient because of less number of network layers being used. Indeed, the combination of GIST features, 512-300-300-1500 DDBN structure, and FEPCD sampling shows a balance between accuracy and computing time. It is also observed that the use of scene-centric dataset (i.e., VGG16-Places365) instead object-centric dataset (i.e., VGG16-ImageNet) shows a better performance in the scene classification task.

7.5 Conclusion

Classifying indoor environments such as homes and offices is not an evident task because of their ambiguity, variability, and scale conditions that robots may encounter during their navigation. We have focused in this chapter on indoor environment classification using global visual features extracted from the GIST descriptor. We use those features for training the Discriminative Deep Belief Network (DDBN) classifier. DDBN is constructed from the Discriminative Restricted Boltzmann Machines (DRBM), which is based on Restricted Boltzmann Machine (RBM) and the joint distribution model. The experimental results clearly ensure that the proposed algorithm can classify indoor environments with almost the same accuracy as CNN, but outperforms it in terms of computational efficiency.

In future work, we will attempt to embed our algorithm in a mobile robot in order for it to recognize its environment, move and act on it. In the next chapter, we will also develop a new approach using 3D sensors and multimodal deep learning for RGBD scene classification.

Chapter 8

Multimodal Feature Fusion for Robust RGBD Indoor Scene Classification

"Fusion has not been proven to be safe, and it is too costly."

Masatoshi Koshiba

Contents

SCENE classification is a challenging problem, especially for indoor scenes due to the wide differences in spatial layouts within each scene class. Recently, with the advent of new 3D sensing technologies such as a Kinect camera, it becomes possible to provide high quality synchronized RGB and depth images (i.e., RGBD data) which together can improve scene classification. Also, after the success of object recognition using CNN on large-scale object-centric datasets, scene-centric dataset called Places was provided for reinvestigating the performance of CNN in scene classification. In this chapter, we propose a new multimodal feature fusion for robust RGBD indoor scene classification. Our architecture consists of two separate CNN trained on RGB and depth images, then combined with a late fusion network. Thereby, we introduce a simple depth colorization method in order to use depth images as inputs for the CNN. Then, we learn RGB and colorized depth images separately using pre-trained CNN on Places dataset, followed by a third training step in which, the architecture fine-tunes two modalities with a fusion network that performs the final classification. In summary, the main contributions are:

- we suggest a multimodal feature fusion for RGBD scene classification;
- we propose a new simple depth colorization method based on polynomial scaling to increase the classification accuracy;
- we combine RGB and colorized depth modalities with a fusion network that performs the final classification.

8.1 Introduction

The creation of autonomous mobile robots requires development in fields such as navigation, scene understanding [127], grasping or scene manipulation. Scene classification is an important research area in computer vision and robotics. It receives vast attention for several practical applications, such as place recognition [270], semantic recognition [30], path-planning [254], and image annotation [171]. Most of the scene classification algorithms are based on features, shapes, and BoWs. The first category is based on the low-level features or global features such as color, texture, and shape. The HSV color histogram [231] and color moment [164] are usually employed thanks to their scale, rotation, and perspective invariance. Oliva and Torralba [177] also introduced texture and shape features in order to extract the edge information in an image scene. Whereas, the second category utilized the local features that model visual features on interest points or regions. A large family of local feature detector is based on local differential geometry such as Harris-Stephens keypoint extractor [73], the Scale Invariant Feature Operator (SFOP) [59], the Harris-Laplace [162], the Harris-Affine extractor, the Maximally Stable Regions (MSERs) [152], and the Scale Invariant Feature Transform (SIFT) [141] which is the successful algorithm for feature detection. The Speeded-Up Robust Features (SURF) [15] is based on the same steps and principles of SIFT detector, but it utilized a different scheme and provided faster results than those obtained with SIFT extractor. All these approaches designed hand-crafted features and lack of capturing high-level information. Recently, features extracted from deep learning methods, particularly those extracted from Convolutional Neural Networks (CNNs) have produced state-of-the-art results for several computer vision tasks, which induced researchers to use CNN learned features for scene classification. After the success of object recognition approaches using CNN trained on large-scale object-centric datasets such as ImageNet, a scene-centric datasets named as Places205 and Places365 were introduced and could be directly used for performing scene classification.

With the arrival of RGBD perception sensors like the Microsoft Kinect, the classification and recognition became fundamental tasks of computer vision research. Initially, the Microsoft XBOX is used in its first application for real-time human pose recognition. Thereafter, its applicability is extended on various works of both robotics and computer vision fields such as object detection and classification [153], SLAM registration and camera pose estimation [229], segmentation [190], body or face tracking [227], and scene classification [237].

Indoor scene classification remains largely unsolved in computer vision and robotics because of the variations of the illumination conditions provided by the only RGB information. To deal with this problem, several works proposed to combine RGB with depth information in order to take the advantage of the RGB image that provides appearance information as well as the depth image that is invariant to the illumination, color, and rotation angle. Moreover, RGB images are directly used as inputs for the CNN, while depth data requires additional processing steps to be suitable for the CNN inputs. The network trained on Places dataset has been trained to recognize places/scenes that follow a specific input distribution (i.e., 3 channels R,G, and B). Whereas, the data coming from a depth sensor describes qualitative features which appear also in RGB images such as edges, shaded regions, and corners, but follows a grayscale rendering of depth data (i.e., 1 channel). However, one solution to adapt depth data to CNN inputs is to use depth colorizing technique in order to convert 1 channel distribution to 3 channels.

In this chapter, we propose a new approach for scene classification for RGBD data. Specifically, we reinvestigated the classical CNN which have shown remarkable performance for recognition on RGB images, in the domain of RGBD data. Encoding depth images in three dimensional form as RGB channels provides for applying pre-trained models already trained for RGB images. We make the statistics of the depth channel similar to the RGB channels. As we are using a pre-trained model, the performance will be the best if the colorized depth image obtained from processing depth image is similar to RGB image as per statistics.

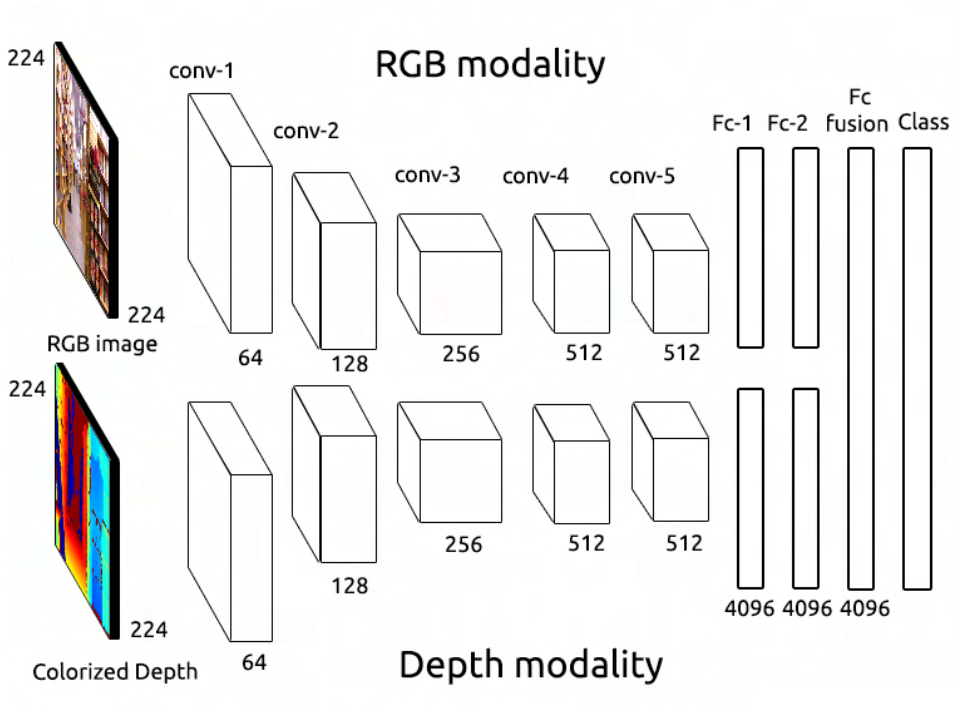


Figure 8.1: Multimodal feature fusion framework for RGBD indoor scene recognition.

As shown in Figure 8.1, our architecture is composed of two CNN modalities which operate on RGB and colorized depth separately. Then it automatically learns to combine these two modalities in a late fusion approach. First, we simply encode a depth image as a rendered RGB image, spreading the information contained in the depth data over all three RGB channels. After that, the architecture initializes both the RGB and depth CNN with weights from a standard pre-trained VGG16 that was trained on Places356 dataset. Finally, each modality is trained separately, followed by a third training stage in which the two modalities are jointly fine-tuned, together with a fusion network that performs the final classfication.

8.2 Depth encoding

Depth images taken from 3D sensors contain the information about the scene geometry. Encoding depth images in three dimensional form as RGB channels provides more rich information about a scene geometry than directly using depth images. There exist several approaches for encoding depth images to three channels. Gupta *et al.* [70] proposed HHA method which encodes the properties of geocentric pose that emphasize complementary discontinuities in the image. The HHA representation encodes for each pixel the horizontal disparity, the height above ground, and the pixel wise angle between a surface normal and the gravity direction. An approach proposed by Eitel *et al.* [50] applied depth colorization using JET colormap. First, they normalized all depth values to be between 0 and 255. Then they applied a JET colormap on the given image which transforms the input from a single to a three channel image (i.e., colorized depth). However, the normalization step with respect to the maximum and minimum values of an image is suboptimal. To handle this limitation, Madai *et al.* [144] computed a standard score for every raw depth value. Then they applied a clipping function, constricting the range of values between -1.5 and 1.5 to detect outliers. A recursive median filter is applied on missing values in order to only consider non-missing values in its kernel. Finally, to find better depth colorization, the authors calculated and colorized the surface normals for every pixel, which better represent the form and the surface structure.

Our method differs from the previous ones by using a simple method of depth encoding that relies on simplified pre-processing steps. The most naive way could be just replicating depth images in the three channels but in our work, we add another step to tweak intensity distribution. First, we normalize depth image values to lie between 0 and 255. Then we apply the simplest non-linear transform i.e., polynomial transform to update the intensity distribution in the depth image. Finally, we assign different colors to the new depth values using JET colormap provided by OpenCV library. This step transforms the depth values from a single channel to colorized depth with three channels. For each pixel $p_k(i, j)$ in the depth image d of size W × H, we map the lowest value to the blue channel and the highest one to the red channel. The value in the middle is mapped to green and the intermediate values are arranged accordingly. Indeed, the colorized depth exploits the full RGB spectrum and can be used as CNN input. This method which encodes depth images in three dimensional form as RGB channels provides more rich information about a scene geometry than directly using depth images.

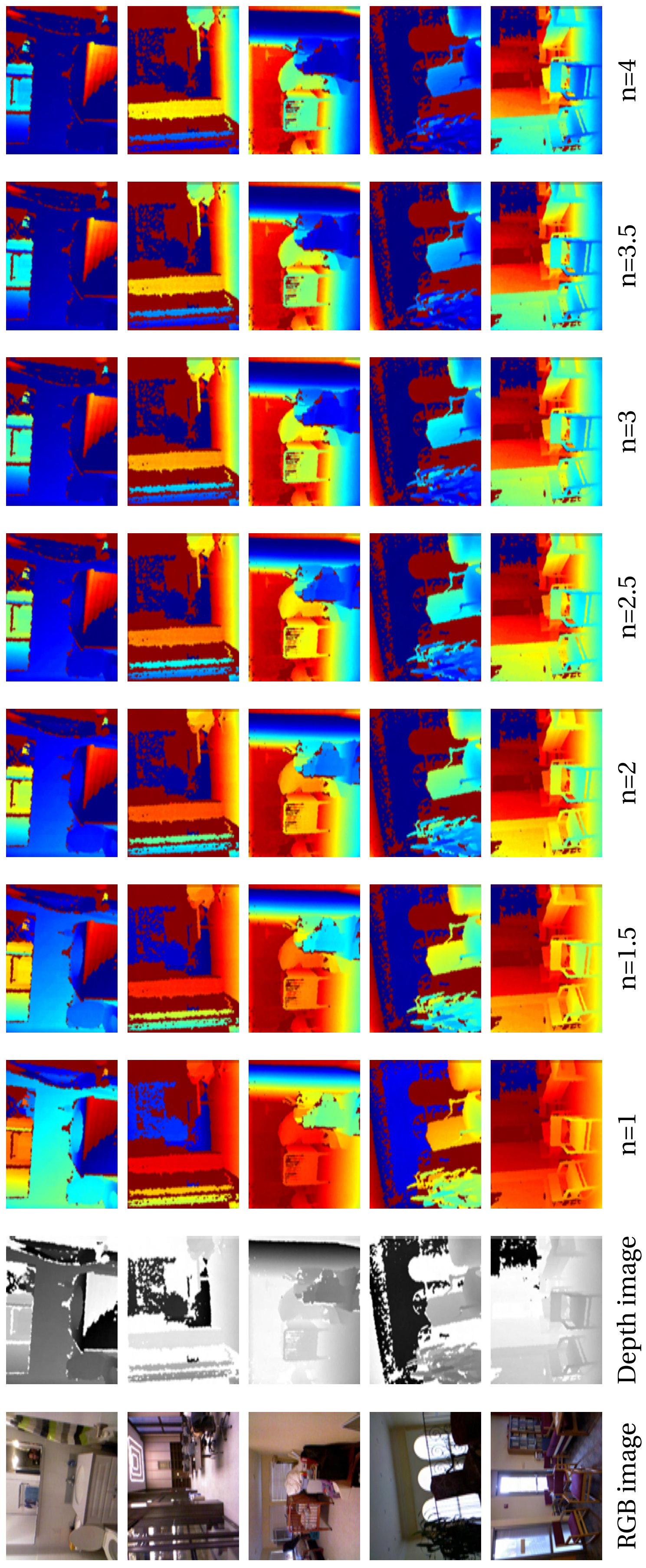


Figure 8.2: Depth encoding method using different polynom degrees (e.g., from 1 to 4).

8.3 Convolutional Neural Network

Convolutional Neural Network is one type of the feedforward neural network. It represents an efficient recognition algorithm which is widely used in pattern recognition and image processing. A CNN is composed of one or more convolutional layers, often with a subsampling layer, which are followed by one or more fully-connected layers as in a standard neural network. By backpropagating the gradients of errors, the network allows learning this multi-stage feature hierarchy (more details are provided in Chapter 2, Section 2.2.4).

8.3.1 CNN architectures

LeNet-5 (1998)

LeNet-5 was proposed by the inventor of the CNN Yann LeCun in 1998 [122]. This network was developed to classify hand-written numbers digitized in 32×32 pixel grayscale images. It consists of only few layers and few filters, because of the computer limitations at that time.

AlexNet (2012)

Alex Krizhevsky and other collaborators designed AlexNet a deeper and much wider version of LeNet-5 [105]. AlexNet is composed of 5 convolution layers, 3 max-pooling 2×2 layers and fully-connected layers. The network was selected in the ImageNet competition that was devoted to the classification of one million of color images onto 1000 classes.

ZFNet (2013)

In 2013, convolutional network from Matthew Zeiler and Rob Fergus called as ZFNet was the winner in ILSVRC 2013 competition [277]. ZFNet was an improvement on AlexNet by tweaking the hyper-parameters of AlexNet while maintaining the same structure with additional deep learning elements.

GoogleNet/Inception (2014)

GoogleNet is the winner of the ILSVRC 2014 competition [233]. It is not only composed on successive convolution and pooling layers, but also on new modules called Inception. GoogleNet is based on several very small convolutions in order to drastically reduce the number of parameters. It is composed of 22 layers. The advantage of this network is reducing the number of parameters from 60 million (AlexNet) to 4 million.

VGGNet (2014)

VGGNet is the runner-up at the ILSVRC 2014 competition [222]. It consists of 16 convolutional layers and is very appealing because of its very uniform architecture. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. Its main contribution was in showing that the depth of the network is a critical component for good performance.

ResNet (2015)

At the ILSVRC 2015, the so-called Residual Neural Network (ResNet) proposed by Kaiming He and other collaborators was introduced [76]. The idea behind ResNet is to add a connection linking the input of a layer (or a set of layers) with its output. These connections are also known as gated units or gated recurrent units and have a strong similarity to recent successful elements applied in RNNs.

DenseNet (2016)

Recently published by Gao Huang *et al.*, the Densely Connected Convolutional Network has each layer directly connected to every other layer in a feedforward fashion [86]. DenseNet obtained significant improvements over previous state-of-the-art architectures on five highly competitive object recognition benchmark tasks.

8.3.2 Training methods

In practice, there exist two ways to train CNN: (1) training an entire network from scratch (i.e., with random initialization) and (2) using pre-trained CNN as an initialization or a fixed feature extractor for the task of interest.

From scratch

In general, all the network parameters are randomly initialized. Layer-sequential unit-variance (LSUV) [166] initialization is a simple method for weight initialization for deep learning methods. LSUV consists of two major steps. Firstly, pre-initialize weights of each convolution or inner-product layer with orthonormal matrixes. Secondly, proceed from the first to the final layer, normalizing the variance of the output of each layer to be equal to one. To quantify the capacity of the network to approximate the ground truth labels for all training inputs, a loss function taking as inputs the weights, biases, and examples from the training set is defined. The most efficient way to find the weights and biases, regarding the number of parameters, is to use an algorithm similar to the Stochastic Gradient Descent (SGD). For each example, the prediction and its associated loss are computed and the loss of each example is summed to compute the final error. Then the backpropagation algorithm is used to propagate the error in order to compute the partial derivatives $\frac{\delta \mathrm{E}}{\delta w}$ and $\frac{\delta \mathrm{E}}{\delta b}$ of the cost function E for all weights w and bias b. Once all the derivatives are computed, the parameters are updated using a chosen optimization technique such as SGD.

Transfer learning

Transfer learning can be mainly divided into two scenarios:

1. features extraction: in this scenario, consider a CNN pre-trained on large dataset (e.g., ImageNet or Places), the last fully-connected layer is removed, then the rest of the network is treated as feature extractor. Finally, a classifier is trained and tested on the features. Typically, the later is a SVM with a linear kernel.

2. fine-tuning the CNN: the second scenario consists in training a pre-trained network on a small dataset. Typically, the last fully-connected layers, which can be viewed as classification layers are reinitialized. Then a small learning rate (in general the learning rate should be small than the one used in training the CNN in large datasets) is applied

to the pre-trained layers. The goal of this strategy is to adapt the features to the new dataset.

8.3.3 Fine-tuning CNN

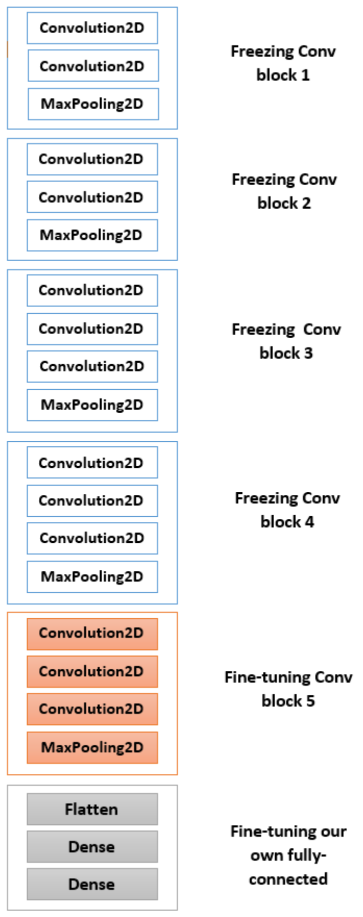


Figure 8.3: VGG16 architecture: fine-tuning the top layers of a a pre-trained network.

In practice, CNNs are trained from scratch (i.e., with random initialization) on large datasets using speed GPU devices. Instead, it is common to pre-train CNNs on a very large dataset, and then use the resulting weights and architecture on other classification tasks using only the CPU device and small data.

For this purpose, we used the pre-trained VGG16 model that was trained on the Places365 dataset (Section 7.4.4). First, we initialize the convolution blocks of the VGG16 model [222]: Conv block 1 with 64 output filters, Conv block 2 with 128 output filters, Conv block 3 with 256 output filters, Conv block 4 with 512 output filters, and Conv block 5 with 512 output filters. Then, we remove the last fully-connected layer whose layer's outputs are the 365 class scores for Places365.

As shown in Figure 8.3, we add our previously defined fully-connected model on top and load its weights (i.e., random initialization). After that, we freeze the layers of the VGG16 model up to the last convolutional block (Conv block 5). In order to improve the classification results, we only "fine-tune" the last convolutional block of the VGG16 model (Conv block 5) rather than the entire network to better fit our data. Since the features learned by low-level convolution blocks are more general, we choose to only fine-tune the last convolutional block which provides more specialized features. Then, we re-train the model on our dataset using small weight updates.

8.4 Multimodal Convolutional Neural Network

Consider $\mathrm{D}_{rgb} = \{x^l, y^l\}$ and $\mathrm{D}_{depth} = \{d^l, y^l\}$ the labeled RGB and depth data for training our multimodal CNNs, where x^i and d^i correspond to the RGB and depth images respectively. Since the number of each modality images are equivalent, the label vector is also equivalent. y^i denoting the image label in one hot-encoding where $y^i \in \mathbb{R}^{\mathrm{L}}$ represents a vector of dimensionality L (i.e., the number of labels) with $y^i_k = 1$ corresponds to the image label for the postition k. Our multimodal CNN is composed of two independent modalities, $\mathcal{N}_{rgb}$ for RGB modality and $\mathcal{N}_{depth}$ for the depth one which are combined $\mathcal{N}_{rgb} \oplus \mathcal{N}_{depth}$ in the fusion stage. Each modality consists of CNN that has been pre-trained for scene classification on the Places365 dataset. Firstly, we initialize the convolution blocks of the VGG16 model, then we remove the last fully-connected layer whose layer's outputs are the 365 class scores for Places365. The key advantage behind starting from this pre-trained CNN is to enable training large CNN with millions of parameters using our limited training data. After that, we add our previously defined fully-connected model on top and load randomly its weights. Then, we freeze the layers of the VGG16 model up to Conv block 5 followed by a fine-tuning step. Moreover, we re-train each modality separately in our $\mathrm{D}_{rgb} = \{x^l, y^l\}$ and $\mathrm{D}_{depth} = \{d^l, y^l\}$ data using small weight updates. Finally, we combined $\mathcal{N}_{rgb}$ and $\mathcal{N}_{rgb}$ modalities into fusion network $\mathcal{N}_{rgb} \oplus \mathcal{N}_{depth}$ that is fine-tuned for multimodal classification of the target data.

Algorithm 5 Multimodal Convolutional Neural Network

```
1) RGB modality: $\mathcal{N}_{rgb}$
    Use pre-trained CNN on Places365
        Initialize the convolution block of the VGG16 model
        Remove the last FC and add new FC layer
        Freeze the layers of the VGG16 model up to Conv block 5
    Prepare a new set of RGB training data $\mathrm{D}_{rgb} = \{x^l, y^l\}$
        Fine-tune Conv block 5
        Re-train the model on $\mathrm{D}_{rgb} = \{x^l, y^l\}$
2) Depth modality: $\mathcal{N}_{depth}$
    Use pre-trained CNN on Places365
        Initialize the convolution block of the VGG16 model
        Remove the last FC and add new FC layer
        Freeze the layers of the VGG16 model up to Conv block 5
    Prepare a new set of RGB training data $\mathrm{D}_{depth} = \{d^l, y^l\}$
        Fine-tune Conv block 5
        Re-train the model on $\mathrm{D}_{depth} = \{d^l, y^l\}$
3) Multimodal CNNs: $\mathcal{N}_{rgb} \oplus \mathcal{N}_{depth}$
    Concatenate $\mathcal{N}_{rgb}$ and $\mathcal{N}_{depth}$ modalities
        Fine-tune the combined network
```

8.5 Experimental results

In this section, we evaluate the performance of our multimodal RGBD scene classification on both NYU V1 and NYU V2 datasets. Then we compare our proposed method with the state-of-the-art approaches that used the same benchmarks.

8.5.1 RGBD dataset

NYU V1

The NYU V1 dataset [218], shown in 8.4 is collected by the New York University. It was captured from a range indoor environements such as residential apartments, workplace and university campus settings. Depth images are acquired by Microsoft Kinect camera, which fulfilled the empty regions and smooth the noise by using the cross-bilateral filter. NYU V1 contains 2347 pairs of images spread over 64 different indoor environments which are grouped into seven categories, including bathroom, bedroom,bookstore, cafe, kitchen, livingroom, and office.

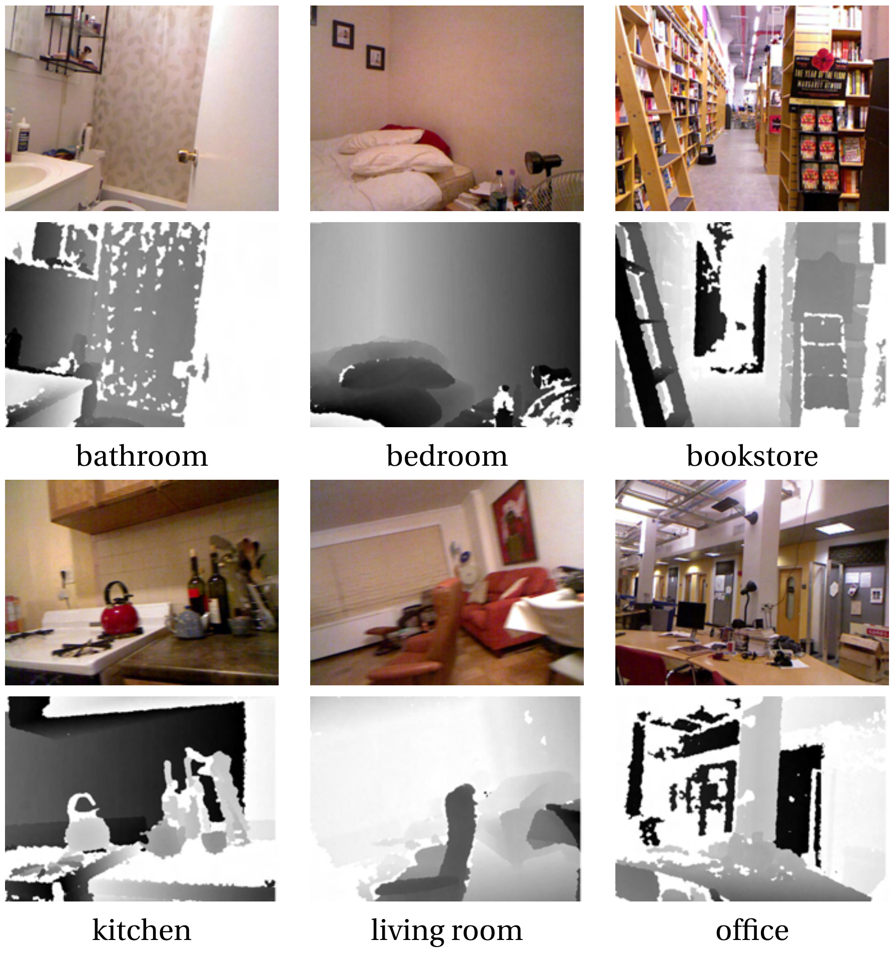


Figure 8.4: Sample images from NYU V1 dataset. Each pair repressents color image and its corresponding depth image shown in grayscale.

NYU V2

Since NYU V1 has limited diversity of scenes, Silberman *et al.* [219] provided a large dataset called NYU V2. It consists of 1449 RGBD image pairs from 464 different scenes, gathered from a wide range of commercial and residential buildings in three different United States cities, comprising 464 different indoor scenes across 26 scene classes. The original 27 categories are reorganized into 10 scene categories, including the 9 most common categories and an "other" category for images in the remaining categories. Figure 8.5 depicts the 9 most common categories which include bathroom, bedroom, bookstore, classroom, dining room, home office, kitchen living room, and office kitchen.

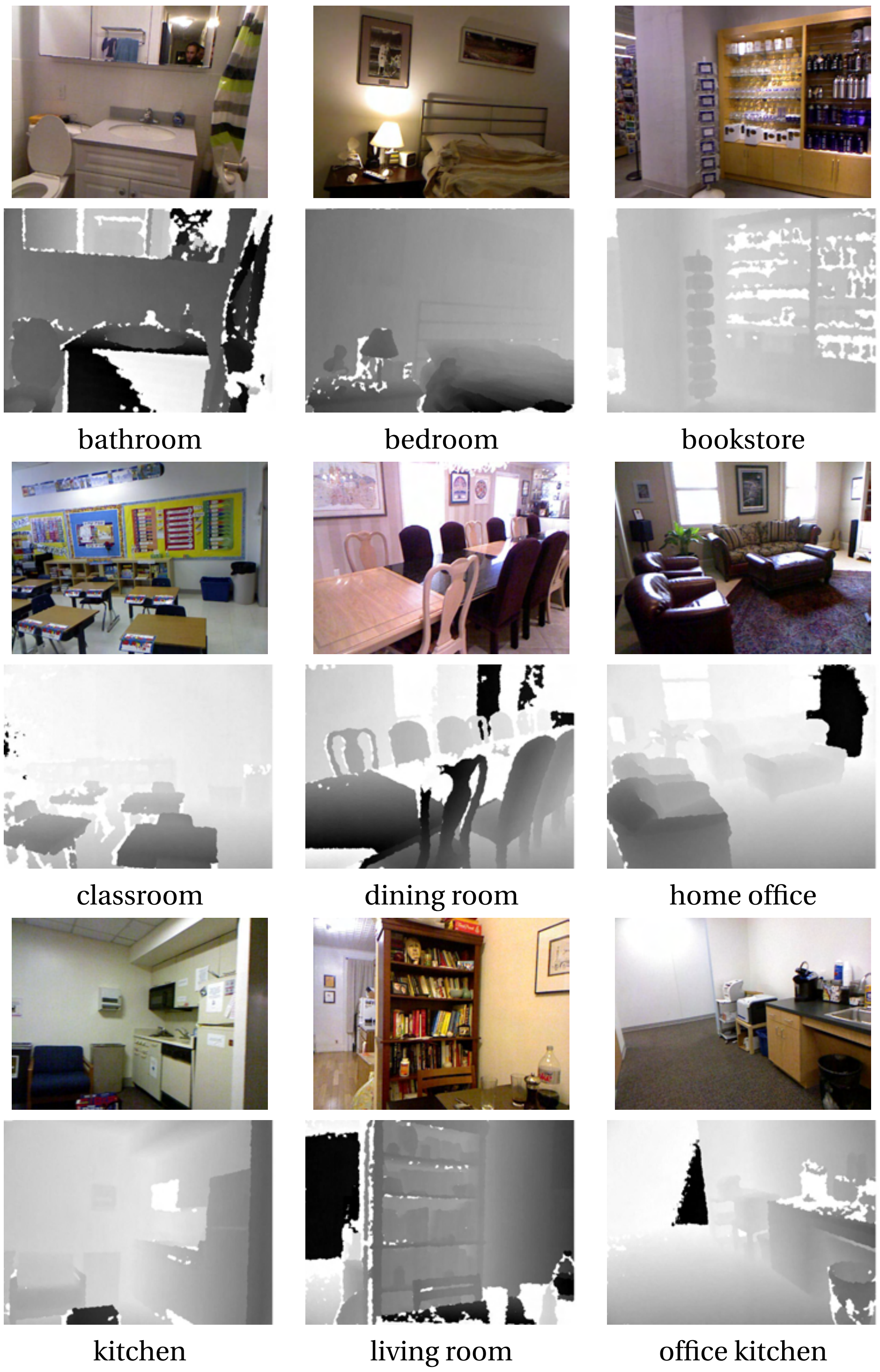


Figure 8.5: Sample images from NYU V2 dataset.

8.5.2 Experimental setup

All experiments are performed using the publicly available Keras framework with TensorFlow as backend. In the image pre-processing step, we apply our encoding approach on depth images using polynomial transformation from degree 1 to degree 4 with step value 0.5. Then we reshape both colorized depth and RGB images into size 224 × 224 to be compliant to the standard input of CNNs. Each modality weights are initialized with the values of a VGG16 model pre-trained on the Places365 dataset. We then fine-tuned each modality separately, using the stochastic gradient descent (SGD) with 0.9 momentum and 0.0001 weight decay. The learning rate is set to 0.001 and decays by a factor of 0.9. After that, we use the similar parameters to fine-tune the fusion netowrk. Table 8.1 summarizes the general characteristics used in our multimodal CNN experiments.

Characteristics	**Values**
Optimizer	Stochastic Gradient Descent (SGD)
Learning rate	0.001
Learning rate decay	1e-6
Momentum	0.9
Loss	categorical cross entropy
Batch size	64
Dropout	0.5
Epochs	1000

Table 8.1: Keras characteristics used in our multimodal CNN experiments.

8.5.3 RGBD scene classification

Our RGBD scene classification consists of two major steps: 1) depth encoding and 2) multimodal feature fusion (i.e., RGB and colorized depth). Before combining the colorized depth images with the RGB ones, we first evaluate the polynomial transformation of depth values that gives the highest accuracy in scene classification using only colorized depth images. For that, we test the polynomial transformation in the range $n = [1.5, 4]$ with the step 0.5. And the polynomial transform n can be easily learned while training. We mention that $n = 1$ corresponds to the basic method of depth encoding that directly assigns colors to the initial depth values.

Modality	**NYU V1**	**NYU V2**
RGB	**90.47** %	**83.33**%
Colorized depth (n=1)	85.71 %	72.33 %
Colorized depth (n=1.5)	88.43 %	74.33 %
Colorized depth (n=2)	89.61 %	78.33 %
Colorized depth (n=2.5)	**89.79**%	**81**%
Colorized depth (n=3)	87.07 %	77.66 %
Colorized depth (n=3.5)	86.39 %	76.66%
Colorized depth (n=4)	84.35 %	75%
Multimodal	**93.19%**	**85.33%**

Table 8.2: Accuracy values of RGBD scene classification evaluated on NYU V1 and V2.

According to Table 8.2, the highest accuracy value of colorized depth classification is obtained with $n = 2.5$. This polynomial transformation provides a suitable representation of colorized depth scenes which can be used for learning robust CNN features. After that, we combined the RGB and colorized depth modalities to improve the RGBD scene classification that reaches the value 93.19 % on NYU V1 dataset and 85.33% on NYU V2 dataset.

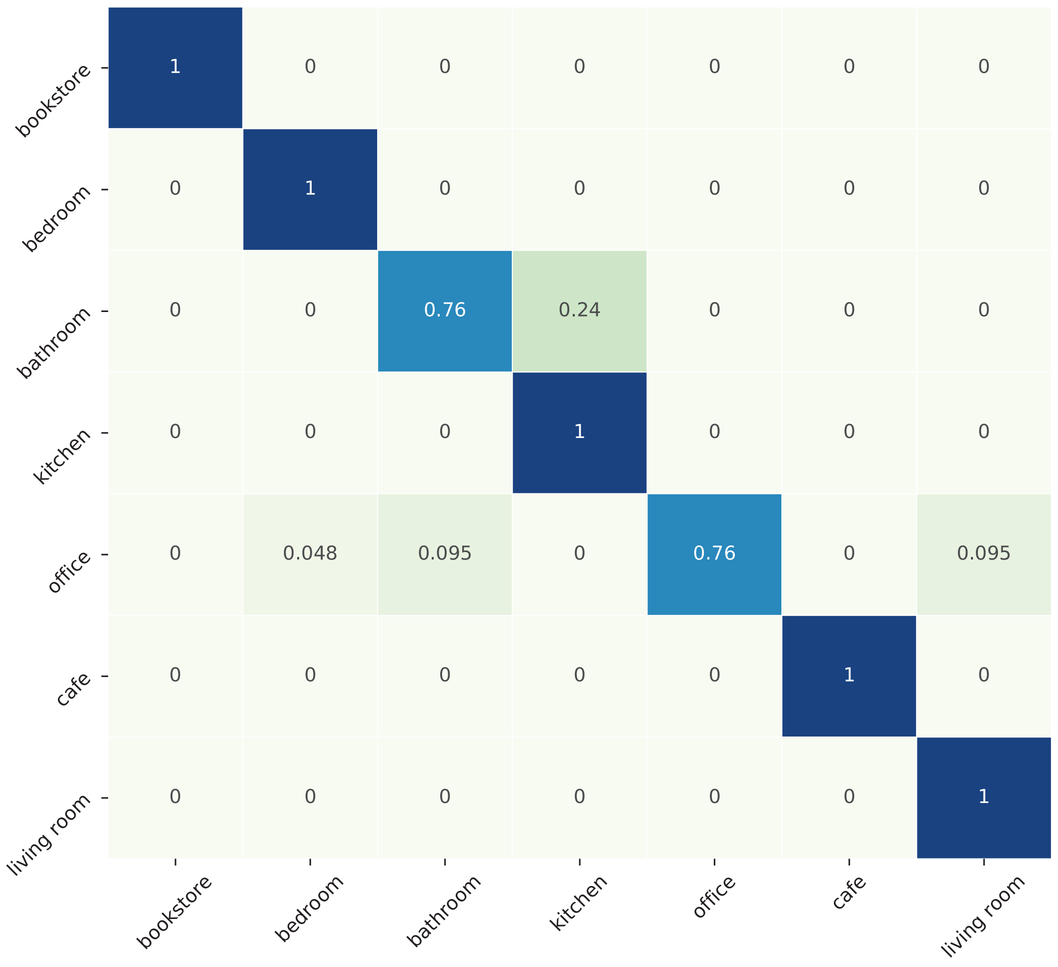


Figure 8.6: Confusion matrix of our multiomodal approach on NYU V1.

Figure 8.6 depicts the multimodal classification confusion matrix evaluated on NYU V1. The average classification rate for classes are listed along the diagonal. The confusions occur between bathroom and kitchen, because there are very similar things in these two scenes, such as the bathroom basins and vanity tops as well as the kitchen countertop. From the confusion matrix we can also find that office is confused with bedroom, bathroom, and living room.

As shown in Figure 8.7, the confusion between classes becomes more important when their number increases. On NYU V2 dataset, dining room class is the most confused class because it contains chairs and tables that can be confused by the sofa in home offices, and the bed in the bedroom. Also, bathroom class is misclassified with the classroom class since the bathroom basins has a similar shape to the student desk.

Tables 8.3 and 8.4 depict the classification reports of our multimodal approach on NYU

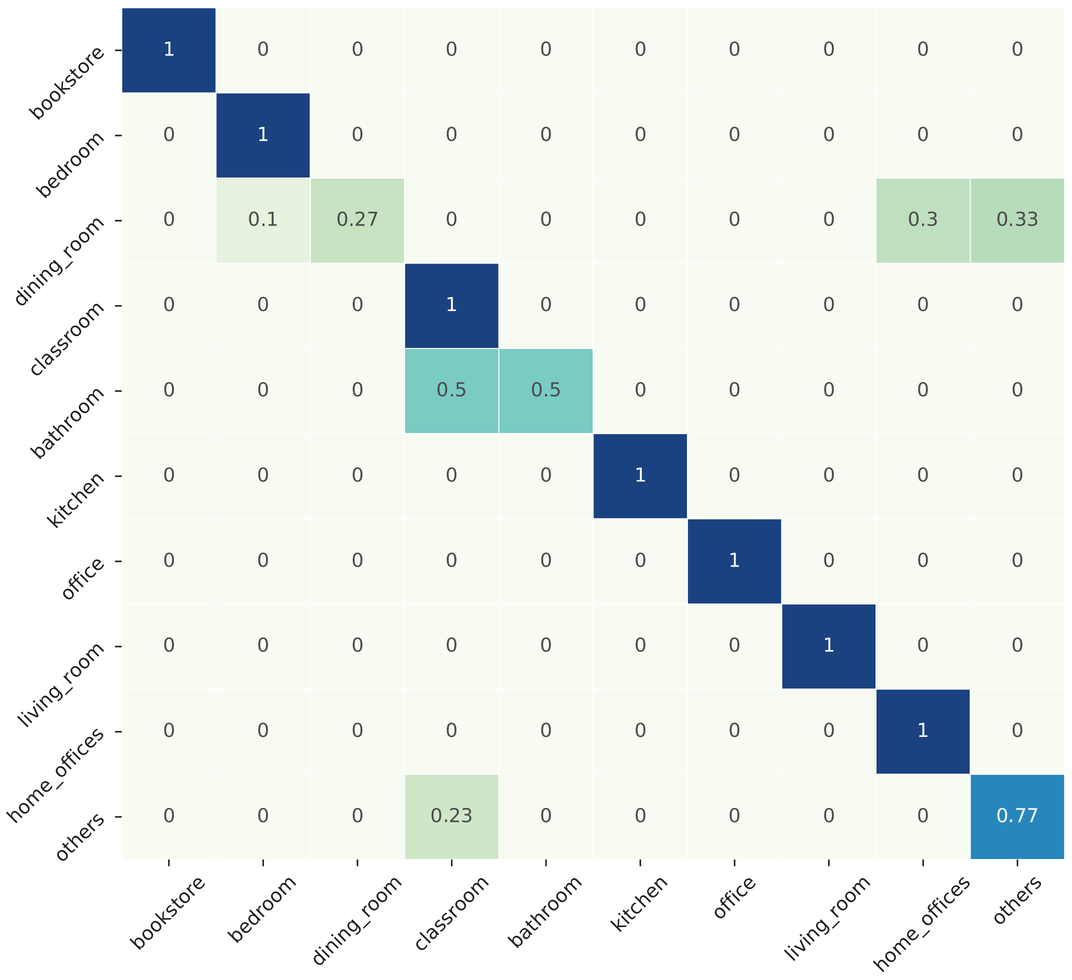


Figure 8.7: Confusion matrix of our multiomodal approach on NYU V2.

Classes	**Classification report**			
	confused classes	**precision**	**recall**	**f1-score**
bookstore	- -	100 %	100 %	100 %
bedroom	- -	95 %	100%	98 %
bathroom	kitchen	89 %	76 %	82 %
kitchen	- -	81%	100 %	89 %
office	bedroom, bathroom, living room	100 %	76 %	86 %
cafe	- -	100%	100 %	100 %
living room	- -	91%	100%	95%
Average	- -	**94%**	**93 %**	**93%**

Table 8.3: The performance of our multimodal approach on NYU V1 dataset.

V1 and NYU V2. The classification report displays a representation of the main classification metrics on a per-class basis. In general, our multimodal approach performs on both NYU V1 and NYU V2 datasets and only few classes which are misclassified. Also, some classes are totally recognized (i.e., the values of precision, recall, and f1-score reach 100%) including, home offices, bookstore, bedroom, kitchen, living room, and classroom.

Classes	**Classification report**			
	confused classes	**precision**	**recall**	**f1-score**
bookstore	- -	100%	100 %	100%
bedroom	- -	91%	100%	95 %
dining room	bedroom, home offices, others	100 %	27 %	42 %
classroom	- -	58 %	100 %	73 %
bathroom	classroom	100%	50%	67%
kitchen	- -	100%	100%	100%
office	- -	100 %	100 %	100 %
living room	- -	100%	100 %	100%
home offices	- -	77%	100%	87%
others	classroom	70 %	77 %	73 %
Average	- -	**90%**	**85%**	**84%**

Table 8.4: The performance of our multimodal approach on NYU V2 dataset.

8.5.4 Comparison to other methods

Table 8.5 shows comparisons with the state-of-the-art methods [54, 68, 74, 102, 228, 237, 261]. Gupta *et al.* [68] proposed a scene classification approach based on the semantic segmentation maps, spatial pyramid, and SVM. While Hayat *et al.* [74] suggested spatial layout and scale invariant convolutional activations. Khan *et al.* [102] exploited rich mid-level convolutional features which are extracted from uniformly and densely image patches using Deep CNNs. Then, the extracted mid-level features are encoded in terms of their association with the codebooks of Scene Representative Patches (SRPs). These approaches worked only on RGB images and obtained 58%, 81.2%, and 80.6% respectively. In our scene classification, using only RGB images, we obtained the value 90.47 % on NYU V1 and 83.33% on NYU V2. This result shows that using pre-trained CNNs which are trained on Places365 dataset provided a good performance in scene classification using only RGB images as CNN input. In RGBD scene classification, Tao *et al.* [237] suggested a Rank Preserving Sparse Learning (RPSL) which takes into consideration four aspects: the first aspect maintained the rank order information of the within-class samples in a local patch while it ignored the rank order information of the between-class samples, the second aspect maximized the margin for the between-class samples on a local patch, the third aspect introduced the L1-norm penalty to obtain the sparse representation, and the final aspect modeled the classification error minimization that used the least squares error minimization. Feng *et al.* [54] suggested Discriminative Locality Alignment Network (DLANet) that adopted the PCANet structure which learns the convolutional filter bank through PCA in order to learn the local features. Wang *et al.* [261] investigated a framework that allows greater spatial flexibility, in which the Fisher vector (FV) encoded distribution of local CNN features. These features are extracted from an augmented pixel-wise representation comprising multiple modalities of RGB, HHA, and surface normals to capture more information of the geometry. Song *et al.* [228] focused on the bottom layers and proposed an alternative strategy to learn depth features combining local weakly supervised training from patches followed by global fine-tuning with images. In our approach, we reinvestigated the classical CNN which have shown remarkable performance for recognition on RGB images, in the domain of RGBD data. For that, we simply encode a depth image as a rendered RGB image using polynomial transformation to update the intensity distribution in the depth image. After that, the architecture initialized both the RGB and depth CNNs with weights from a standard pre-trained VGG16 that was trained on Places356 dataset. Finally, each modality is trained separately, followed by a third training stage in which the two modalities are jointly fine-tuned, together with a fusion network that performed the final classfication. We conclude that our depth encoding approach provided pertinent colorized depth images, thus increasing the classification rates 93.19 % on NYU V1 and 85.33 % on NYU V2 when combining with RGB images into multimodal fusion feature approach.

Approaches	**NYU V1**			**NYU V2**		
	RGB	Depth	Both	RGB	Depth	Both
Gupta *et al.* [68]	- -	- -	- -	58%	- -	- -
RGBD-LLC [237]	78.1%	68.5%	79.9%	- -	- -	- -
DLANet [54]	80.33 %	71.43 %	82.66 %	- -	- -	- -
S^2ICA [74]	81.2%	- -	- -	- -	- -	- -
DUCA [102]	80.6%	- -	- -	- -	- -	- -
Combined FV and Full [261]	- -	- -	- -	53.5 %	51.5%	63.9%
RGB-D-CNN [228]	- -	- -	- -	53.4%	56.4%	65.8%
Our appraoch	90.47 %	89.79%	**93.19 %**	83.33%	81%	**85.33 %**

Table 8.5: Comparison of state-of-the-art and our approach accuracies.

8.6 Conclusion

In this chapter, we proposed a novel multimodal neural network architecture for RGBD indoor scene classification based on a simple depth encoding approach and pre-trained CNNs. First, we provided an effective depth encoding method which used polynomial transformation of depth values and JET colormap technique. Second, we combined RGB and colorized depth modalities with a fusion network to improve the scene classification task. Finally, we carried a set of experiments and achieved the state-of-the-art performance on the NYU V1 and NYU V2 datasets. In future work, we will investigate our multimodal approach in mobile robotic navigation applications which used recently scene classification to determine the robot location.

Part III

Topological navigation

Chapter 9

Topological Navigation: literature review

"The idea that a robot will become more aware of its environment, that telling it to 'go to the kitchen' means something - navigation and understanding of the environment is a robot problem. Those are the technological frontiers of the robotics industry."

Colin Angle

Contents

AUTONOMOUS robots operating in indoor environment settings must be able to navigate in large, dynamic, and unknown spaces. They frequently should answer three questions: "where am I?", "where are other places relative to me?", and "how do I get to other places from here?" This chapter answers these questions by introducing a concise definition of navigation, its strategies, and different environment mapping. Then the literature review of the field is introduced.

9.1 Mobile robotic navigation

Human navigation is a basic and critical ability that is responsible for daily human activities. It appears that humans have a very adequate ability to recognize places, to remember locations, to travel long distances, and to find their way out. They navigate through a novel environment by drawing on their previous experiences in similar conditions. Moreover, they reason about obstacles, free-space, and the topology of the environment, guided by common sense rules for navigation. Animals like humans have also important navigation abilities. They spend much of their time moving from one place to another, learn and return to the specific place, and take the suboptimal path in the mathematical sense that can rapidly be selected. In the context of animal navigation, Gallistel [61] defined the navigation as "the process of determining and maintaining a course or trajectory from one place to another. Processes for estimating one's position with respect to the known world are fundamental to it. The known world is composed of the surfaces whose locations relative to one another are represented on a map." This definition consists of two hypotheses: 1) that the world in which the animal moves and lives is represented within the brain, and 2) that this representation can be named *a map*. In autonomous navigation, a robot asks frequently three questions during its navigation in the environment: 1) "where am I?", 2) "where am I going?", and 3) "how do I get there?" In order to tackle these questions, the robot has to:

- build a map of its environment (i.e., mapping);
- self localize itself in the environment (i.e., localization);
- plan a path from its location to the desired location (i.e., path planning).

Therefore the mobile robot requires to model the environment in the form of a map based on the environment characteristics and the robot sensors. Then, it finds its position in an environment based on its representation. Finally, it defines a path in a map from one place to another in order to reach its goal. Depending on the map type, it is possible to follow different strategies for path planning.

9.1.1 Navigation strategies

The navigation strategies that allow a robot to move towards a goal are enormously diverse. We introduce the classification established by Trullier *et al.* [251], in which, they divided the navigation strategies into two families: without internal models (i.e., local strategies) and with internal models (i.e., global strategies).

Local strategies

- Target approaching: in this strategy, the robot can move towards a visible object from its current position based generally on the perception of the object. This strategy uses reflex actions, in which each perception is directly associated with an action. It is a local strategy because it is functional only in the area of the environment in which the goal is visible.

- Guidance: this strategy achieves a goal that is not a material object that is directly visible, but a point of space characterized by the spatial configuration of a set of remarkable objects or landmarks, which surround it or which are close to it. The principle is to move in a direction in order to reproduce this configuration. This ability seems to be used by some insects (e.g., bees) and has been reproduced on various robots. This strategy also uses reflex actions and performs local navigation that requires the landmarks. Finally, it should be noted that this type of navigation does not require any spatial modeling. Indeed, the mapping of a perceptual memory with the current perceptions does not require in any case a specific spatial treatment.

Global strategies

- Place recognition-triggered response: this is the first ability to perform global navigation. It is possible for a robot to reach a goal from positions for which the landmarks that characterize its location are invisible. It is generally applied to large environments and is based on the concept of place. This strategy requires an internal representation of the environment which consists of defining places as areas of space in which perceptions remain similar and associating an action to each of these places. The sequence of these actions defines a path that leads to the goal. These models provide significant autonomy. A path that joins a goal cannot be used to reach another different goal, thus the strategy will lead to learning a new path.

- Metric navigation: it uses a geometric representation of the world, which can be delivered by a user, or built by the robot itself. It allows the robot to plan paths within unexplored areas of its environment. For this purpose, it memorizes the metric positions relating to the different places, in addition to the passage from one place to the other. These relative positions make it possible, by the simple composition of vectors, to calculate a trajectory going from one place to another even if the possibility of this displacement has not been memorized in the form of edge/link.

- Topological navigation: this ability makes it possible to memorize in the internal model the spatial relationships between the different places in the environment. Indeed, these relationships indicate the possibility of moving from one place to another, without associating them with a particular goal. With this strategy, the internal model calculates different paths between two random places. However, this model only allows the planning of movements among the known places and along the known paths.

Through navigation strategies, we can conclude that global methods guarantee safety and optimization in navigation. Nevertheless, local methods bring speed and control of the visible dynamic environment. The main limitations of global methods are the need of prior modeling of the environment and the limitation on the number of degrees of freedom of the system under consideration. On the other hand, these so-called global methods guarantee the solutions from which the choice of a particular trajectory can be made according to constraints on trajectory shape or an optimization criterion.

9.1.2 Robotic mapping

The robotic mapping consists of acquiring spatial models of physical environments through mobile robots. The mapping process is generally regarded as one of the most important problems in the autonomous mobile robotic. Thrun [241] defined robotic mapping as the processing of sensor inputs performed by an autonomous robot to:

1. create a map of its environment, and then
2. localize itself within this map.

The term mapping is used to describe only the actual creation of the map, while the term localization is used to describe the determination of the robot's position within the map. The field of mapping was widely divided into metric, topological, and hybrid approaches. Metric maps capture the geometric properties of the environment. Topological maps describe the connectivity of different places. Hybrid maps combine the advantages of both metric and topological maps. The robot often encounters some mapping problems that generally result from the following reasons:

- map size: the size of the map depends directly on the size of the navigation environment, the larger it is, the more the processing is heavy and the storage space is important.
- noise: actuator and sensor measurements may be noisy due to external or internal agents in the robot. It is essential to apply a treatment in this case to reduce these problems.
- data association: when the different places have the same aspects, it is complex to establish correspondences between them during the exploration.

Metric maps

Metric maps are very useful for small scale path planning and obstacle avoidance (Figure 9.2a). Due to the correct geometry, metric maps created by robots have the advantage of being easily readable by human operators. In such maps, the environment is represented by a set of objects with associated positions in a metric space, usually in two dimensions. This space is mostly the one in which the position of the robot estimated by the proprioceptive data is expressed. Using a metric model of the sensors, perceptions allow detecting these objects and estimating their position relative to the robot. The position of these objects in the environment is then calculated using the estimated robot position. In some locations, these objects correspond to the obstacles that the robot may encounter in its environment. The environmental map then corresponds directly to the free space, in which the robot can move. Primarily, two methods are developed for storing information in the form of a metric map. The first is to explicitly extract objects from perceptions and save them in the map with their estimated position. The second is to directly represent the free space accessible to the robot and the areas of obstacles that it can not cross. The most common type of metric map generated by mobile robots is the second method so-called *occupancy grid map* in which the world is divided into discrete cells that can be marked as either unknown, empty, or occupied. As depicted in Figure 9.1, the environment is fully discretized according to a regular grid with a very fine spatial resolution. A probability of occupation is associated with each element of the grid. This probability measures the confidence that the corresponding space in the environment is actually occupied by an obstacle. This probabilistic method is usually adopted so that the current map at all times is the most probable given the measurements.

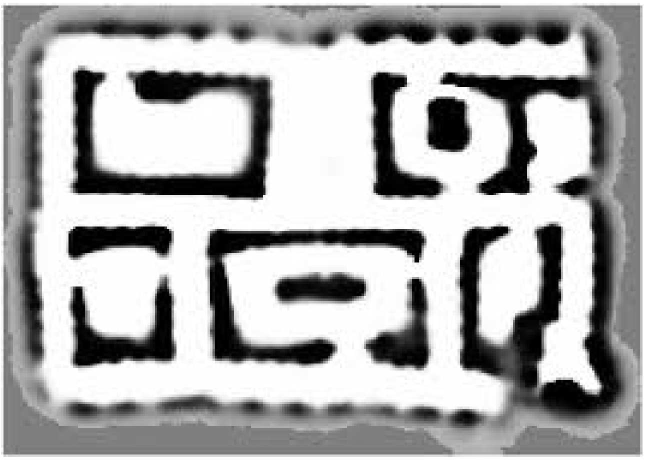


Figure 9.1: Occupancy grid map. Dark areas indicate a high probability of an obstacle [239].

Topological maps

According to Thrun [240], the first topological map appeared in the late 1970s in the work of Kuipers who modeled the world as a graph of places. More commonly, topological maps can be viewed as abstract representations that describe the links between elements in the environment without using an absolute reference frame. As shown in Figure 9.2b, the nodes of the graph correspond to places. While the edges linking these nodes mark the transition from one place to another and memorize in general, the manner of making this transition. Topological maps are useful for path planning and localization. Localization is typically done by checking neighboring nodes against the current node to check whether a transition between nodes has occurred. A topological map can store some geometric information in the links between nodes. In this case, path-planning is a straight-forward task.

The nodes in the map determine the whole process of building the topological map. However, the choice of what may represent the nodes usually depends on the robot perception which must be able to detect the places in question. The detection of these places can be constrained by the choices of a human operator or be completely autonomous. The first possibility to define the nodes is designed by the constructor. It defines directly the places that must be detected by the robot. Procedures are then written, allowing to specifically detect each type of place. The most common choice is the use of corridors, doors, and intersections. On the other hand, instead of defining completely the places that can detect the robot, the constructor can simply define the situations where the robot can record a place, leaving it to define each place at the time of the discovery. For instance, the constructor can provide the robot with the general ability to detect a door or window. When the robot detects such an object, it will record a new node in the map. The last method of building the topological map is to define the nodes as areas where the perceptions are approximately constant. In general, this result is obtained by unsupervised categorization of perceptions. Those are grouped into categories containing similar data, without specifying the categories by the constructor.

The edges represent the link between the nodes. They memorize data on the neighborhood relations between the places represented by the nodes. These data are generally obtained through proprioceptive information. They can be more or less precise and represented in various forms. The adjacency information between two places represented by connected nodes is the first information that an edge can represent. The existence of an edge means that the robot can move directly from one place to another, without moving through an intermediate place. Edges can also store metric information about the relative position to places. This method has the advantage of limiting the accumulation of the proprioceptive data errors since they are only used on the connecting distance between nodes. It is also possible to associate a position with each node in order to integrate the proprioceptive data with a topological map. This method measures the proprioceptive data in the space in which is expressed and corresponds to the position of the different places in the environment. We could deduce that this type of map is very close to the metric maps, with the difference that only the places visited by the robot are memorized, and not the objects perceived by the robot.

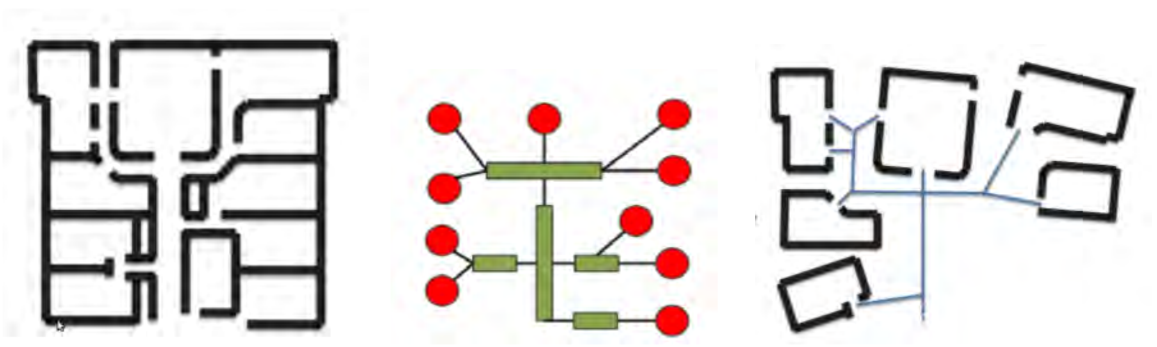
Table 9.1 and Figure 9.2 show the comparison of metric and topological maps. According to this comparative study, we can conclude that the use of topological maps will be very useful in mobile robotic navigation.

Hybrid maps

Topo-metric or hybrid maps (Figure 9.2c) aim at combining the advantages of both topological and geometric maps. Topological maps identify the robot location relative to the model based on different sensor features and establish the important information for global navigation. However, every node in a topological map may represent a large area, where the stated information could not be enough to compute trajectories to operate the robot inside a node. Hybrid maps combine both representations, where the topological component is located at a high-level of abstraction and each node contains a distinct and independent geometric map. This type of representation is for example well suited to construct maps from the vision sensors in which each node of the graph can be associated with an image, connected to its neighbors by information obtained by the odometry of the robot or by visual odometry. The main idea under hybrid map building is composed of two steps:

1. the definition of the topological map as a set of nodes and links to accomplish the mission;

2. each node in the map has an associated geometric map which has a coordinate system (i.e., a global/local referential providing the parametric information to the local path planning and obstacle avoidance).



(a) Metric map (b) Topological map (c) Hybrid map

Figure 9.2: Different representations of the environment.

	Topological map	**Metric map**
Advantages	- No metric model of sensors to merge proprioceptive data and perceptions within a unified representation of the environment; - Memorizing the environment in the form of a set of distinct places: more direct definition of places related to the perceptual capacities of the robot; - Use of less concepts: the topological map is closer to the possibilities of the robot, by memorizing its perceptions and its movements; - Discretization of the environment corresponding to the choice of places represented in the map; - Avoid geometric approximations by primitives; - Structure exploited by motion planning algorithms; - Reasonable memory space.	- Representation of the entire environment, and not a small subset of places: accurately and continuously estimate the position of the robot on the whole of its environment; - The position of the robot is unambiguously defined by its coordinates within the space in which the map is represented; - Use high-level concepts such as objects, obstacles, or walls.
Drawbacks	- No estimation of data for unvisited positions: topological maps generally require a very complete exploration of the environment to represent it accurately; - Difficulty of recognizing places when the sensors are noisy, or environments are very dynamic.	- Difficulty in obtaining a metric model of the sensors; - Complexity of the calculation of the path within a metric map: the progress of the planning in a continuous space and not in a discretized space; - Necessity of a large memory space to store the measurements in the map.

Table 9.1: Advantages and drawbacks of both topological and metric maps.

9.2 State-of-the-art

Topological mapping methods aim to segment the environment by creating topological places linked together on the topological graph. In this review, we divided topological navigation approaches into six families: approaches which are robust to perceptual aliasing, probabilistic-based approaches, appearance-based approaches, place recognition-based approaches, memory-based approaches, and finally deep learning-based approaches.

9.2.1 Robust approaches to perceptual aliasing

Perceptual aliasing is firstly introduced by Whitehead and Ballard in 1991 [265]. It designated a situation where "a state in the world, depending upon the configuration of the sensory-motor subsystem, may map to several internal states; [and] conversely, a single internal state may represent multiple world states." This situation is very common when robots use distance sensors to avoid obstacles such as ultrasonic sensors. Such sensors are, for example, able to measure the position of the robot with respect to a corner, but they don't provide any information on the position along a corridor. If the algorithm is not robust to perceptual aliasing, it will be difficult to determine exactly where the robot is located.

Goedemé *et al.* [64] also suggested a construction of the topological map in the same manner. The detection of a loop closing is based on a mathematical concept derived from the theory of evidence called the Dempster-Shafer theory. This theory offers an alternative to traditional probabilistic theory for the mathematical representation of uncertainty. The advantage of this method is that it makes a distinction between multiple types of uncertainty and defines two types of uncertainty: aleatory uncertainty, and epistemic uncertainty. The aleatory uncertainty represented the type of uncertainty which resulted from the fact that a system can behave in random ways. Whereas, the epistemic uncertainty represented the type of uncertainty which resulted from the lack of knowledge about a system. For topological map building, a series of omnidirectional images are acquired then clustered into places. After that, loop closing hypotheses are formulated between similar places of which evidence is collected using Dempster-Shafer theory. Cummins and Newman [43] proposed a probabilistic framework for navigation and mapping named FAB-MAP. FAB-MAP allowed to explicitly account for perceptual aliasing in the environment by learning a generative model of place appearance. New place models are learned online from only a single observation of place by partitioning the learning problem into two parts. Given a visual place, the FAB-MAP system calculated the probability that the place matches any previously visited location, as well as the probability that the place is from an unvisited location. Visual places, and hence locations in the real-world, can be associated with high probability matches in appearance space. The complexity of the algorithm is linear in the number of places in the map and is particularly suitable for online loop closing detection in mobile robotics. Finally, the authors evaluated FAB-MAP using New College and City Center datasets and showed that the system is robust even in visually repetitive environments and is fast enough for online loop closing detection. Angeli *et al.* [10] used the Bayesian filtering to calculate the detection of the loop closing by encoding the images in the shape of a set of local primitives in concordance with the paradigm of the Bag of Visual Words. If a hypothesis to detect a loop closing receives a high probability, an algorithm of the geometry of multi-views is used. The solution is completely incremental, and it allows for the detection of a loop closing in real-time. Also, the authors integrated odometric information from a mobile robot with the visual topological SLAM to obtain globally consistent maps, and by adapting the framework to achieve global localization. Experiments were conducted using a Pioneer 3 DX mobile robot from MobileRobots Inc. equipped with an onboard camera which provided images of size 320×240 pixels. During this experiments, 7 loop closing were correctly detected. Liu and Zhang [139] provided a simple method for visual loop closing detection in appearance-based SLAM. They used direct feature matching to improve the recall of loop closing detection and therefore avoid the perceptual aliasing problem caused by the vector quantization process of BoWs. Feature matching is a time-consuming problem if the linear search is used. In fact, direct feature matching with linear search can become intractable especially when the number of images increased. For this reason, the authors used a kd-tree to index visual features and achieve a fast match, followed by a verification step confirming a true loop closing.

9.2.2 Probabilistic-based approaches

Several approaches have investigated Bayesian methods for probabilistic navigation. Simmons and Koenig [220] used a Partially Observable Markov Decision Process (POMDP) to estimate the possible locations of the robot in the form of probability distribution. This model is constructed from topological information about the connectivity of the environment, approximate distance information, and actuator characteristics. The Markov model estimated the robot's position in the form of probability distributions, which are updated when the robot has moved or turned, and when it observed features such as walls and corridor junctions. A planner associated a directive (e.g., turn or stop) with every Markov state in order to guide the robot's behavior. Whenever the probability distribution of the Markov model is updated, the total probability mass for each directive is calculated, and the robot executed the one with the largest probability mass. The approach presented many advantages including the ability to account for uncertainty in the robot's initial position, sensor noise, actuator uncertainty, and uncertainty in the sensor data interpretation. Also, by integrating topological and metric information, the approach processed with uncertainty arising from incomplete descriptions of the environment. Tomatis *et al.* [244] adopted the approach presented by [220] in a hybrid system that combined topological and metric models for both localization and map building. The metric model consisted of infinite lines that belong to the same place. These places are related to each other by means of a topological map which is composed of nodes representing topological locations and edges between nodes. The system used a 360° laser scanner to extract corners and openings for the topological approach and lines for the metric method. A global topological map connected local metric maps, allowing a compact environment model, without requiring global metric consistency and allowed both precision and robustness. Also, the approach dealt loops in the environment during automatic mapping by means of the information of the multimodal topological localization. Cheng *et al.* [40] suggested a topological map building-based localization and navigation method. They considered nodes as the vertices such as intersections and links as the corridors connecting these vertices. The map generation is based on the corridor classification problem and solved by a progressive Bayesian classifier. The authors extracted features from multi-observations, then fused them to achieve more robust performance. The generation of the topological map as well as loop closing are proposed to build the environment map through mobile autonomous exploration. Using the derived map and the Markov localization method, the robot can then localize itself and navigate in the indoor environment. Also, experiments showed that the method can be deployed in a mobile robot since it can learn the map directly and autonomously using very low memory and computational requirements.

9.2.3 Appearance-based approaches

In a topological Simultaneous Localization and Mapping (SLAM), Valgren *et al.* [256] used the algorithm Incremental Spectral Clustering (ISC) in order to gather omnidirectional images in clusters. A matrix of affinities highlighted similarities between the current image and all the clusters of images. Then, the ISC algorithm determined the optimal number of clusters according to the matrix entries depending on the appearance only. The method is well suited to the problem of appearance-based, on-line topological mapping for mobile robots. The authors showed that the environment-dependent parameters of the clustering algorithm can be reduced to just a single, intuitive parameter. Also, experimental results in large outdoor and indoor environments showed that loop closing is correctly detected by computing only a fraction of the entries in the affinity matrix.

Several studies have been made based on omnidirectional images, allowing for the recognition of a place from a distant viewing point. Kumar *et al.* [108] made use of decision trees Extremely Randomized Trees (ERT) to detect a loop closing. They used visual primitives extracted from the omnidirectional images to produce Bag of Visual Words (BoVWs). A vocabulary tree is built offline. Then, descriptors from each sequence image are added sequentially to the vocabulary tree using inverted files. When the vehicle is close to a previously visited location, the descriptors of the current image are used in conjunction with the inverted files in order to obtain the *n* closest images. After that, geometric consistency based on the five-point algorithm with preemptive RANSAC to get a fast estimate of the number of inliers in the epipolar geometry is used to determine if the matches are good. The approach is evaluated using a Pointgrey Ladybug 2 camera mounted on a vehicle, on a trajectory of 6.5 kilometers. Gaspar *et al.* [62] proposed a method for the visual-based navigation in indoor environments, using a single omnidirectional camera (i.e., catadioptric). The authors presented the geometry of the catadioptric sensor and the method to obtain a bird's eye (i.e., orthographic) view of the ground plane. This representation significantly simplified the solution to navigation problems, by eliminating any perspective effects. Also, they proposed two major navigation modalities: topological navigation, and visual path following. In topological navigation, the robot traveled long distances and does not require knowledge of its exact position but rather, a qualitative position on the topological map. The navigation process combined appearance-based methods as well as visual servoing upon some environmental features and consisted of graphs that describe the topology of the environment. The qualitative position of the robot on the graph is determined efficiently by comparing the robot's current view with previously learned images, using a low-dimensional subspace representation of the input image set. Whereas, in the visual path following, the robot is controlled to follow a pre-specified path accurately, by tracking visual landmarks in bird's eye views of the ground plane. By combining these two navigation modalities, the authors achieved an overall system which exhibits improved robustness, scalability, and simplicity. Fazl-Ersi *et al.* [53] proposed hierarchical classifiers for topological robot localization. They represented images using the Scale Invariant Feature Transform (SIFT) on a regular dense grid instead of the traditional use of SIFT descriptors. Then they used an agglomerative clustering technique in order to build visual words by grouping visually similar features extracted from the training images. This method ensures that only visually similar patches are grouped together and the resulting clusters are compact. For each image, a spatial pyramid representation is built by repeatedly subdividing it and computing histograms of visual words at increasingly fine resolutions. After that, the authors applied an information maximization technique to build a hierarchical classifier for each class by learning informative features. Finally, they proposed feature hierarchy construction using the redundancy between the features to select for each top-level feature, a set of child features which provide similar information as their parents, complementary to information provided by other top-level features. Pronobis *et al.* [185] proposed a topological robot localization based on the incremental method which performs like the batch algorithm while reducing the memory requirements. They defined a place as a nameable segment of a real-world environment such as a kitchen, an office, and a corridor which are uniquely identifiable because of their specific functionality and/or appearance. The authors combined an incremental extension of SVMs that was introduced in [232] with a method of reducing the number of support vectors required to build the decision function without any loss in performance [49]. The resulting combination achieved the same recognition results as the original incremental method while reducing the memory requirements. Experiments are conducted on two common scenarios: (1) adaptation in presence of dynamic changes, and (2) transfer of knowledge between two robot platforms working in the same task. The results showed clearly the effectiveness of the approach in terms of accuracy, speed, reduced memory, and the capability to forget redundant and outdated information. Chapoulie *et al.* [38] proposed an approach for outdoor and indoor environment segmentation using spherical images. They defined a place as scene whose structure parameters are almost constant. During robot navigation in the same topological place, the structure parameters extracted from the perceived environment remained constant. Environment structure parameters were appearance frequency and orientation of straight lines, textures, curvatures, and repeated patterns. Given this definition, the authors required to estimate environment structure parameters and detected changes in them. Structure parameters are estimated using GIST descriptor. As the spherical representation of the environment is used, GIST descriptor should be adapted to spherical images by removing the zero-padding system along the image horizontal axis and keeping it along the vertical one. In topological map building, the transitions between the topological places are defined related by GIST variations of these places. Whereas, spherical images which presented the same structure parameters are clustered together in a topological place. Then, change-point detection algorithm and Neyman-Pearson lemma are used to detect GIST changes in online and constant time. Finally, the experiments are tested on the Kahn building in the INRIA Sophia-Antipolis research center as well as INRIA Sophia-Antipolis campus environments to prove the validity of the topological place definition.

9.2.4 Place recognition-based approaches

Place recognition has become an important issue for mobile robot applications in recent years. It is mainly used for robot navigation in the indoor environment.

Wang and Lin [262] proposed an encoding method for scene change detection and recognition towards topological map building. The problem of detecting scene change events is performed using an omnidirectional camera mounted on a mobile robot. Then these events are employed to build a topological map for recognizing the nodes of the visual places subsequently. In order to automatically label the areas of a robot environment, the authors used the scene change events to perform this task. They designed a binary code transform named "Hull Census Transform" (HCT) for scene representation, which is in average about 10-30 bit codes for an image frame (i.e., one hull case). HCT consisted of sparse data with respect to image features or images themselves for visual place representation. It is used to handle the scene change conditions and to take the varying environment into account. This characteristic is very helpful for further topological map building and visual place recognition, particularly for the catadioptric vision sensors. The method with repeatedly generating the convex hull from the image features and computing the relative magnitude between image features over the convex hull is fast and robust under illumination change and has shown promising results on the COLD datasets. It enables the mobile robot to recognize scenes based on the image appearance and automatically add nodes into the existing topological map. Lin *et al.* [133] suggested a scene recognition technique related to the human perception for the scene change detection as well as a topological map construction using omnidirectional image sequences. They provided Extended Hull Census Transform (Extended-HCT) semantic scene descriptor which is based on the SURF features. Extended-HCT consisted of the color information of the environment and the structure information of the convex hull feature points which are extracted from the original Hull Census Transform (HCT) method [262]. The visual codewords of Extended-HCT are adapted to the fast moving scenes, especially when exploring the unknown environment. The images are described using six parameters including the relationship between the feature vectors, the structured relation among the feature points in the image, and the color histogram indexing information associated with the environment. Finally, topological map construction is built after the autonomous mobile robot passing through an unknown path and can then be used to assist for the place recognition, localization, and navigation tasks if the robot goes through the environment in the future. The experimental results are presented for both the indoor and outdoor navigation scenarios. Liu et Siegwart [137] described a lightweight scene recognition approach using an adaptive descriptor. The approach consisted of online registration of new scenes onto a topological representation and solved the localization challenge to topological regions simultaneously. The authors proposed a new descriptor based on geometric information and color features which are extracted from an uncalibrated omnidirectional camera. It characterizes a whole image based on the average of the U-V color space values of the pixels enclosed in different areas of the image. Then the authors adopted a Dirichlet Process Mixture Model (DPMM) in order to approximate conditional probabilities of the new measurements providing incrementally estimated reference models. Lin *et al.* [134] presented a vehicle localization method to assist vehicle navigation based on scene recognition and topological map construction. They constructed a topological map in which the node information is used for place recognition and derivation of vehicle location using omnidirectional image sequences. To perform the scene change detection and topological map construction, the authors utilized the SURF descriptor and the Extend-HCT method. Once the keypoints of the omnidirectional images are extracted by the SURF detector, they are described by six parameters (i.e., features of convex hulls, cost, score, color histogram index, average distance of center, and average distance of feature), the relationship among feature magnitude, feature point structure, and finally color information. Then Content-Based Image Retrieval (CBIR) and Feature-Based Image Retrieval (FBIR) methods are combined to perform the image retrieval for localization. The CBIR method is based on color, texture, shape, spatial correspondence, and other information in the images. The Compact Composite Descriptors (CCD) is used to compute the similarity between dataset and the ground truth. This method describes the information of different scenes using lower-level feature points and combines the fuzzy system to classify and index the images. While the FBIR method uses feature matching by SURF and classification by k-nearest neighbors to calculate the descriptor distance of the query and candidate images. Experiments showed that the proposed approach is able to construct a real-time image retrieval system for navigation assistance.

9.2.5 Memory-based approaches

Labbé and Michaud [110] proposed a new approach for online large-scale and a long-term operation called Real-Time Appearance-Based Mapping (RTAB-Map). The algorithm is based on the creation of online signatures of current images stored in a memory. The memory will then assessed the quality of the extracted primitives and the size of the data, and stored those that deserve to be selected in Long-Term Memory (LTM) for a limited time. The old signatures already stored in the LTM will be transferred to Working Memory (WM). The latter contained the ideal place to evaluate the detection of the loop closing using a Bayesian filter. Finally, the authors conducted a set of experiments on Community datasets, Université de Sherbrooke (UdeS) dataset, and "Need for Speed: Most Wanted" (NFSMW) dataset. Erkent and Bozma [52] considered long-term topological place learning and proposed an approach enabling the robot to learn in an unsupervised, organized, and incremental manner. They used only appearance information and places are detected either manually or using a place detection algorithms [16, 39, 95]. The information associated with the previously visited places are internally stored in the form of Bubble Descriptor Semantic Tree (BDST) using the previously proposed bubble space representation. As such, BDST corresponded to long-term place memory, which was generated and maintained without any external supervision. It organized the learned knowledge where the terminal nodes are viewed as corresponding to separate places while its structure encoded their semantic hierarchy. To learn new places, BDST is updated to incorporate new information in an unsupervised and incremental manner. For this purpose, a hierarchical Single LINK clustering (SLINK) algorithm- that is known to achieve theoretical order-of-magnitude bounds for both efficiency of storage and retrieval- is used. Finally, the approach is evaluated on the combined benchmark indoors COLD dataset and outdoors New College dataset in which the robot is able to retain efficiently and use the knowledge associated with the learned places. Karaoğuz and Bozma [96] proposed an autonomous topological spatial cognition approach for mobile robotic navigation. They defined the concept of "places" as a set of appearances or locations sharing common perceptual signatures or physical boundaries. In this approach, as the robot navigates around, places are detected then either recognized or learned along with mapping as necessary. First, it explicitly incorporated a long-term spatial memory in which the knowledge of learned places as well as their spatial relations are retained in place and map memories. In place memory, a tree structure organized the set of learned places in a hierarchy based on their appearance-related similarities. Whereas, in map memory, topological maps stored any observed spatial relations among different places. Second, the robot built its spatial memory in an organized, incremental, and unsupervised manner. It detected places by partitioning the sequence of the associated descriptors based on their coherency. Then, it attempted to recognize each detected place by relating to its long-term spatial memory. In the case of recognition, its already existing knowledge is updated. On the contrary case, place learning and mapping are invoked as to incorporate the new place and its spatial relation.

9.2.6 Deep learning-based approaches

Recently, deep learning techniques have been used for robotic navigation tasks. Wang *et al.* [263] proposed a novel Omnidirectional Convolutional Neural Network (O-CNN) which combined the advantage of both a modern omnidirectional camera configuration and deep learning methods. Consider a robot that is only given a few place exemplars on the map, and its current location is unknown and away from these exemplars. The goal of the O-CNN is to check the closest place exemplary and estimate the relative distance between the input and this closest place. O-CNN presented three important design elements: (1) a circular padding to both image and CNN feature spaces to reflect the fact that omnidirectional images have no true image boundary, (2) a roll branching approach to conquering rotational variation in the captured omnidirectional images, and (3) a modified lifted structured feature embedding to provide the concept of distance in the environments, which was called continuous lifted structured feature embedding. Finally, the authors built a new virtual world framework on Unity 3D, containing several indoor scenes in order to train the O-CNN. Savinov *et al.* [208] introduced a new deep learning-based memory architecture for navigation, inspired by landmark-based navigation in animals. Semi-Parametric Topological Memory (SPTM) is composed of two components: a non-parametric memory graph G where each node corresponds to a location in the environment, and a parametric deep network R capable of retrieving nodes from the graph based on observations. The graph stored only the connectivity of locations corresponding to the nodes without any metric information. In the environment exploration, the agent built the graph by appending observations to it and adding shortcut connections based on detected visual similarities. While the network R is trained to retrieve nodes from the graph based on the environment observation. By this way, the agent is able to localize itself in the graph. Finally, the authors complemented an SPTM-based navigation agent by using the memory with a locomotion network L, which allows the agent to move between nodes in the graph. The R and L networks are trained in self-supervised fashion, without any manual labeling or reward signal. Gupta *et al.* [69] proposed the Cognitive Mapper and Planner (CMP) for visual navigation. CMP consisted of a spatial memory to capture the layout of the environment, and a planner that can plan paths given partial information. A unified architecture combined the mapper and the planner and can be trained to leverage regularities of the environment. The mapper merged information from input views as observed by the agent over time in order to provide a metric egocentric multi-scale belief about the environment in a top-down view. While the planner utilized this multi-scale egocentric belief of the environment to plan paths to the specified goal and outputs the optimal action to take. CMP constructed a top-down belief map of the environment and applied a differentiable neural net planner to produce the next action at each time step. The accumulated belief of the environment allowed the agent the ability to track visited regions of the environment. The planner presented three advantages, 1) it naturally treated with partially observed environments by explicitly learning when and where to explore, 2) it allowed to train the mapper for navigation, and 3) it enabled to plan paths to distant goal locations in time complexity that is logarithmic in the number of steps to the goal.

9.3 Conclusion

This chapter presented an overview of mobile robotic navigation in indoor environments, navigation strategies, different types of maps as well as topological navigation approaches. Metrical maps model an environment in a single metric reference frame, wherein important space portions and entities take place, or just the property of the occupied/free state of the space is represented at high resolution (i.e., occupancy grid maps). Whereas, topological maps model spatial knowledge as a graph, describing locations, and places as nodes, and their spatial relations, such as proximity and links as edges. Due to the advantages of topological maps, we propose in the next chapter a new topological navigation approach based on topological map building and omnidirectional images.

Chapter 10

A Novel Incremental Topological Mapping using Global Visual Features

"The map? I will first make it."

Patrick White

Contents

MAPPING is fundamental in the navigation task of autonomous mobile robots. In appearance-based mapping, the process of detecting visual loop closing determines whether the current observation comes from a previously visited location or a new one. The purpose of this chapter is to present a new method of exploring indoor environments by an autonomous mobile robot [290, 298], as well as building topological maps based on global visual attributes. This method takes advantage of the small size of the GIST descriptors, and the ease of their calculation. We also make use of omnidirectional images to build a single global visual descriptor showing an entire room. Furthermore, in order to handle the problem of a visual loop closing, we have employed a formula that correctly assigns each global descriptor to its location.

Our contribution in this chapter is inspired by the incremental approach as described by [148], which allows for the construction of the topological map from spatial representations extracted from the catadioptric sensor. It also focuses on the construction of the topological map from omnidirectional images, and relies on the following criteria:

- ensure incremental topological mapping of the environment;
- use small-sized global descriptors " GIST " for signature computing;
- calculate a threshold for the detection of a loop closing.

10.1 Introduction

An autonomous mobile robot is supposed to navigate in real, dynamic, and unknown environments. However, if it has no prior information about the environment where it needs to move and act, it must be able to represent it. This model is essential for the robot's localization and its planning of its movements to carry out its missions. In certain robotic tasks, a map of the environment is built incrementally by merging the successive perceptions acquired from the sensors of the robot during its navigation. The first question that arises in this context though is "how to represent an environmental map?"

Traditionally, robot maps have been classified into three categories: metric, topological, and hybrid. Thrun [239] advocated that metric maps are useful for small-scale path-planning and obstacle-avoidance tasks. They contain a geometrically correct representation of the environment. The locations are memorized along with their global coordinates. Topological maps record a set of places accessible to the robot as well as the manner to move from one place to its neighboring ones. They are used to travel long distances in the environment, without demanding accurate control of the robot position along a path. In this type of map, the environment is represented by a graph which segmented the environment into different places and has laid out the connections between them. Nodes in the graph correspond to recognizable places, and links are associated with regions where some environmental structure can be used to control the robot. Hybrid maps use the functions of the two previous ones complementarily by combining local geometric information with more global structural information. In all these maps, it is important for the robot to know when it returns to an already visited place to correct its map and establish new paths. Therefore, when it detects such new paths, it needs to update its map accordingly in order to be able to perform tasks requiring more effective movement.

Omnidirectional vision has been used widely for place recognition and visual topological mapping tasks due to its capability of capturing the rich information (i.e., 360° field of view) from the surrounding environment. In this chapter, we propose new topological navigation based on the advantages of omnidirectional images with a horizontal field of 360° view and the extraction of GIST features. Also, we have proposed a formula that correctly assigns each global descriptor to its location in order to build a coherent topological map that respects the visual loop closing. Figure 10.1 depicts the major steps of our approach:

- capture omnidirectional image from the current place;
- unwrap the omnidirectional image to obtain a panoramic image;
- compute small-sized global descriptors " GIST " for signature computing;
- calculate the similarity measurement using L2-distance;
- update the topological map building following Algorithm 6.

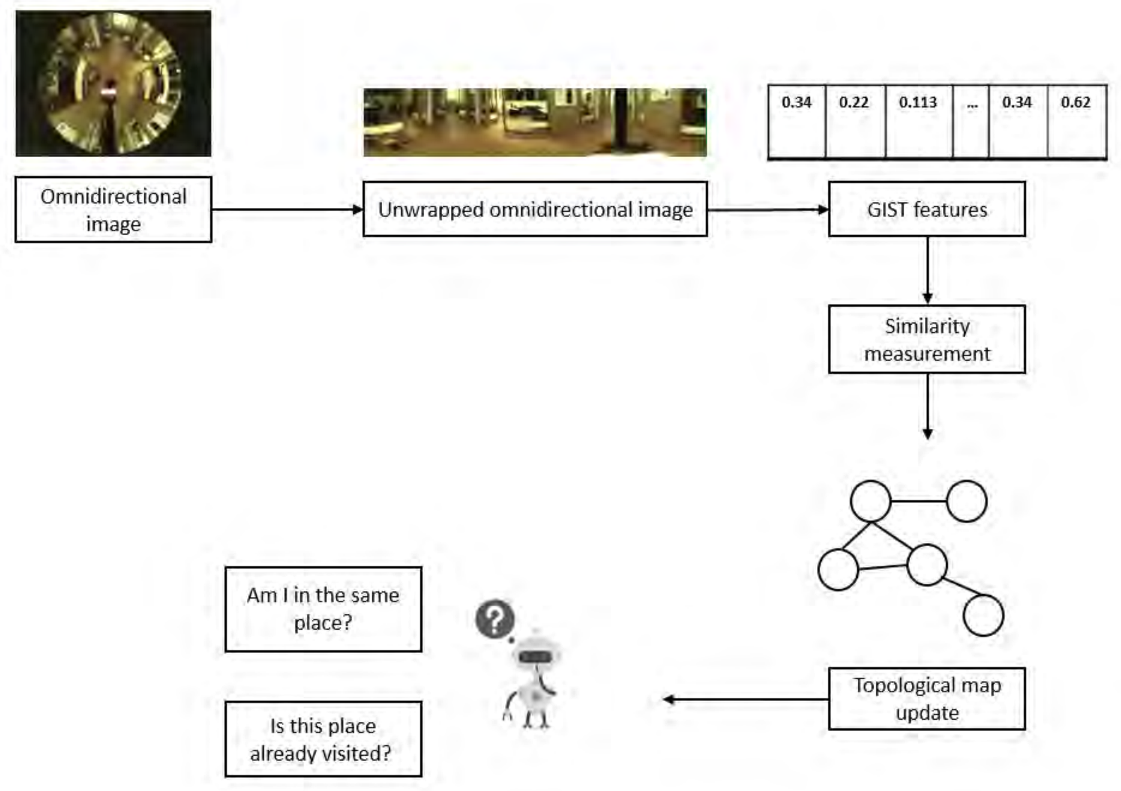


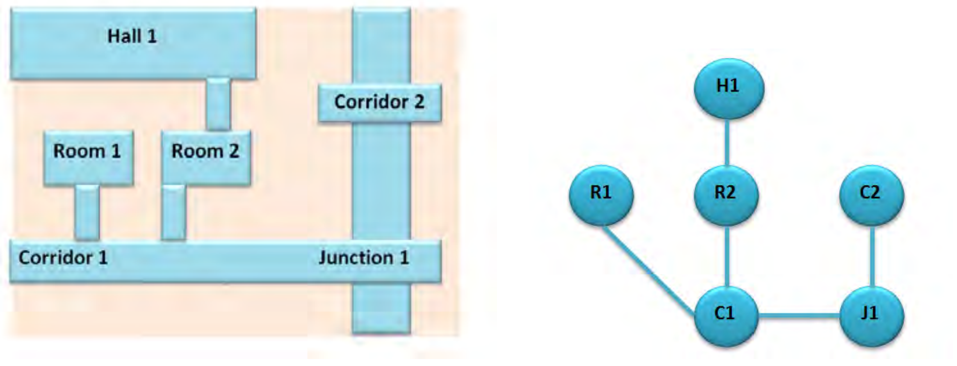
Figure 10.1: An overview of our toplogical navigation approach.

10.2 Environment representation

A robot that has no prior information about the environment in which it must navigate, must be able to model its environment through all of its sensors. This model is essential for the robot to localize itself and to plan its movements in order to accomplish its missions. A mobile robot must be able to perceive its environment and react to unexpected changes. However, the mapping of such a place should be incremental by combining the successive perceptions acquired by the sensory system of the robot during its exploration. Typical maps are categorized as a metric map, topological map and hybrid map. Metric maps utilize metric measurements as the basis of representation. The goal is to achieve complete and precise modeling of the measurable environment in terms of Euclidean distance. Topological maps join locally salient features as nodes and add edges/links between pairs of nodes. The hybrid map is a combination of local metric maps and global topological maps. In this thesis, we are interested in the topological mapping process.

10.2.1 Topological map

A topological map is a high-level representation of the environment. It attempts to capture the spatial connectivity of the environment components by presenting them in a graph of topological features such as rooms, corridors, and intersections. Nodes in such a graph correspond to distinct situations and places and are connected by edges if there exists a direct path between them. The resolution of a map is proportional to the complexity of the environment representation. Compactness, being a key factor in this type of representation, and its low usage of computer processing allow for fast planning and facilitate interfacing with symbolic planners and problem-solvers. However, with no metric information available, the topological representation requires features (or landmark) selection, detection, and recognition. This means that topological representation is heavily dependent on a powerful system to identify key elements of the environment. As a result, one of the most localization problems using topological representations occurs when the robot traverses two places that look alike. The topological mapping often has the difficulty of determining if these places are the same or not, particularly if these places have been reached via different motion commands, actions or paths.



(a) Environment map (b) Topological map

Figure 10.2: Example of topological representation. (a) shows actual map of environment. (b) illustrates topological approach.

Why a topological map ?

The key advantage of topological maps lies in their compactness. Such maps correspond completely to the complexity of the environment. We justify the use of topological maps for the following reasons:

- they keep a record of the environment as a set of distinct places;
- they allow fast planning and low space complexity;
- they discretize the environment components in concordance to the places shown in the map;
- they provide natural interfaces for human instructions such as "*Go to room X!*"

What is a node?

Topological representation has a slightly different meaning depending on the authors. In some work, the authors consider every image or scan of the environment as a node. For approaches to topological mapping that are based on underlying metric maps, they assume that every room should be represented by a node. Hence, "narrow passages" or "portals" in the metric map, are identified as the edges between nodes. When each node is presented as a unique node, it becomes very easy to visually verify the correctness of the topological map. This type of topological map is very well suited for human-robot interaction as well. Once a human operator has labeled the nodes, there will be no ambiguity when the robot is asked to move to a particular location. Kuipers *et al.* [107] described the environment as a collection of places, paths, and regions, linked by topological relations such as connectivity, order, boundary, and containment. A place represented a part of the environment as a zero-dimensional point. A path described a part of the environment, for example, a street in a city, as a one-dimensional subspace. A path may describe an order relation on the places it contains, and it may serve as a boundary for one or more regions. A region represents a two-dimensional subset of the environment. A region may be defined by one or more boundaries, by a common frame of reference, or by its use in an abstraction relation. In outdoor navigation, the natural segmentation of rooms and corridors might not exist. Nevertheless, humans tend to represent the world in topological terms, e.g., in the parking, close to the tree, outside of the school. These semantic labels on the world shift, depending on the task. For instance, it is perfectly valid to ask the question "where on the parking?" The answer might be "next to the cinema", which shows that there exists a hierarchy in the representation.

Our map

A topological map is built to support the navigation of a mobile robot. To enhance representation of the environment, the robot makes use of an omnidirectional camera, and the acquired data are processed aiming at extracting the most relevant features of the environment. The built topological map provides the essential information needed for the navigation process. In our map, we consider the nodes presented in a topological map as places where global visual features are quasi-constant, and each node regroups similar features in a single category. As to edges, they represent a transition from one place to another.

10.2.2 GIST descriptor

The GIST descriptor is based on the extraction of the spatial envelope of the image that contains the essential information. The spatial envelope is a set of holistic scene properties that can be used for inferring the semantic category of the scene without the need of recognition of the objects in the scene. The GIST features are computed by converting the input image to grayscale, normalizing the intensities and locally scaling the contrast. The resulting image is then split into a grid $M \times M$ *cells* into several scales, and the response of each cell is computed using a series of Gabor filters. All of the cell responses are concatenated to form the feature vector, which describes the image in its globality. The GIST descriptors represent the best results with the default setup: filter bank at *4* scales and *8* orientations. These descriptors, being $4\times 8\times 16=512$ dimensional, are computed for grayscale variant. Figure 10.3 depicts GIST descriptors of our unwrapped omnidirectional images.

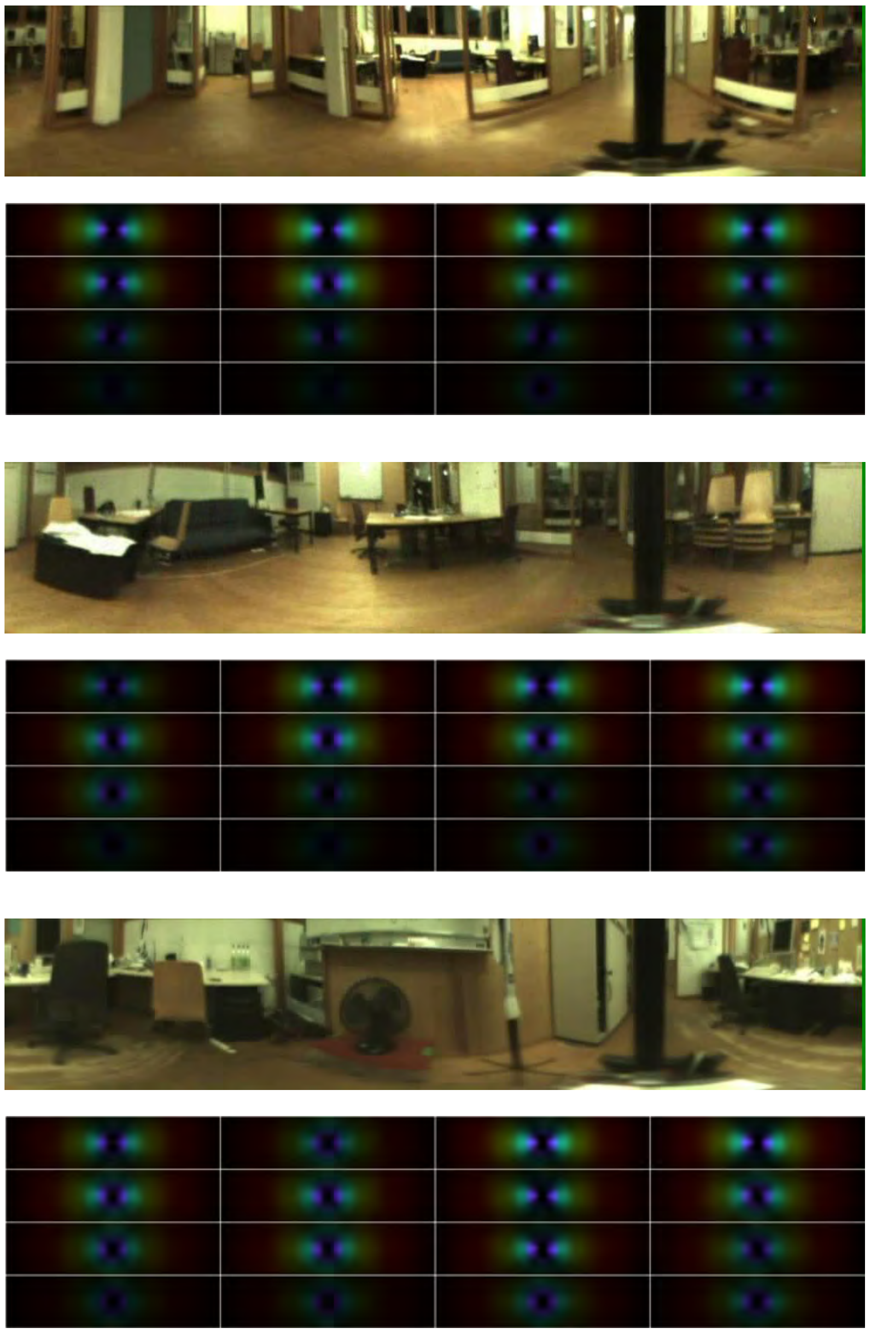


Figure 10.3: GIST descriptors of unwrapped omnidirectional images.

10.2.3 Unwrapping omnidirectional images

Omnidirectional cameras (i.e., from omni, meaning all) is a camera with a 360° field of view in the horizontal plane, or with a visual field that covers a hemisphere or approximately the entire sphere. They provided a 360° view of the robot's environment, into a single image, and have been applied successfully to autonomous navigation. Omnidirectional images are usually obtained with catadioptric panoramic cameras, which combine conventional cameras and convex mirrors. Generally, mirror shapes can be spherical, conic, parabolic or hyperbolic.

Before extracting descriptors for omnidirectional images, we used the Pronobis's software[1] to unwrapper and represent them as panoramic images. This application converts polar to cartesian coordinates. The size and aspect ratio of the panoramic images can be adapted and the application performed bicubic or bilinear interpolation to improve the quality of the produced image. The center of the omnidirectional image is detected automatically using two different methods. The first one is fast and very simple algorithm based on image thresholding. The second is slower but more robust based on Hough transform and edge detection.

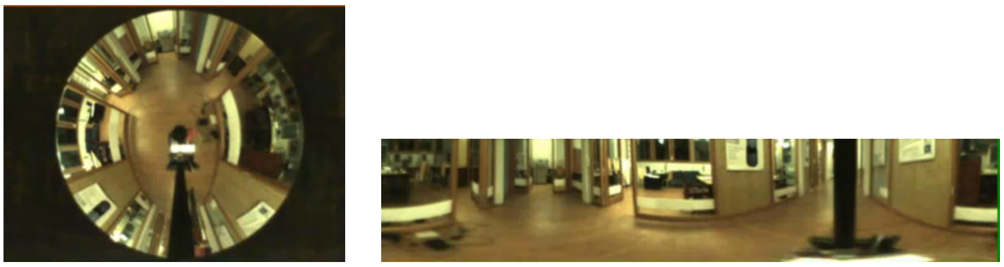

(a) Omnidirectional image (b) Panoramic image

Figure 10.4: Example of unwrapped omnidirectional image (frame 1).

10.2.4 Similarity measurement

The GIST descriptor is applied for each acquired image at instant *t*. Our method takes as input the unwrapped omnidirectional image of fixed size and generates a vector of dimension *512*. The GIST vectors are compared using the L2-distance. In the following, a similarity measurement is exhaustively computed involving the L2-distance between two consecutive images.

$$\mathrm{L2}-distance(t,t+1)=\sum_{n=1}^{512}(gist_{(t)}(n)-gist_{(t+1)}(n))^2 \tag{10.1}$$

Where

- $gist_{(t)}$ = the GIST descriptor of omnidirectional image acquired at instant *t*;
- $gist_{(t+1)}$= the GIST descriptor of omnidirectional image acquired at instant *t+1*;

Equation 10.1 is used to calculate the similarity between two consecutive images. Two images belong to the same place when the distance is minimal.

[1]https://www.pronobis.pro/software/unwrap/

10.3 Detection of loop closing and topological mapping

Marie *et al.* [148] presented an incremental approach that allowed for the construction of a topological map based on spatial representations constructed from:

- the local topology defined by the segmentation of the free space;
- the visual signature constructed by selecting the most pertinent information in an image.

When the robot starts exploring the environment, it should first extract a signature corresponding to the initial position, hence creating the first node of the map. And during the exploration, new signatures are created allowing the robot to relocalize itself and define new places.

Every now and then, the robot must calculate the difference of the current signature with that of the node. For the robot to change the place, this difference should be less than the threshold τ_1. In case the similarity measurement is below this threshold, the robot searches within the ensemble of the nearest matching nodes, assessing this time the similarity between the current signature and the signatures of all other nodes. If the similarity exceeds a second threshold τ_2 ($\tau_2 = 0.8$ in this case), the robot must conclude that this node is already visited, and consequently a loop closing is detected.

In our work, we use a global signature to represent the "gist" of the environment. The advantage of this representation is that it is very compact and fast to compute. In addition, we combine the method of exploration and construction of the topological map described by Marie *et al.* [148] with a method that allows for an automatic calculation of the thresholds of the loop closing during the mobile robot navigation.

10.3.1 Detection of loop closing

While the mobile robot is building a map of an unknown environment under exploration, the place is currently visited may seem similar to one or more encountered earlier. When this is the case, the robot asks naturally the question: "is this place the same I visited earlier, or a new one?" The detection of that place is previously visited by the robot is known in navigation as *loop closing*. In general loop closing detection algorithms can be classified into three families: map-to-map, image-to-map, and image-to-image:

- map-to-map loop closing is established by splitting the global map into sub-maps and then finding correspondences between them [41];
- image-to-map loop closing performs the search of the matches between image and a map [266];
- image-to-image loop closing searches correspondences between the latest image from the camera and the previously seen images [43].

Map-to-map loop closing approach is very intense performance-wise since it treats a large amount of information on each iteration while comparing sub-maps. However, this approach is not efficient in large-scale environments. Image-to-map loop closing approach is fast and accurate, but in practice, it is very memory intensive because one needs to store both point cloud map and all the image features. Finally, the image-to-image loop closing approach is well used in large-scale environments and can be computed fast with feature-based approaches.

Since the detection of the loop closing is crucial in improving the robustness of navigation algorithms, we have accordingly developed a new image-to-image method to deduce whether or not the robot has encountered a place already visited during its exploration of an indoor environment. This method is essentially based on a statistical calculation of the threshold of an image belonging to a specific category of rooms.

Considering a set of images of a specific room, the threshold of each one of those image is defined from the maximum L2-distance between the GIST descriptor of each image and the average of all the GIST descriptors of this set of images. The following equation shows the calculation of the threshold for each room (i).

$$\tau_{loop}(i) = max(distance(\mathrm{GIST}_{(i)} - mean_{(i)})) \tag{10.2}$$

Where

- ***GIST***=matrix of global descriptors for all images stored in the room (node *i*);
- ***mean***=mean of the GIST matrix.

10.3.2 Topological mapping

Initially, when the robot begins the exploration of a given environment, its strategy is to carefully record multiple images (i.e., covering several views) in every discovered room of the laboratory.

At the same time, a signature $S_{t=0}$ that corresponds to the initial position is built, and stored by default in the first node in the map. During the robot's movement, new signatures are created hand-in-hand allowing it to get situated and define new places.

At time *t*, when the robot is on the node *n*, the probability that at time *t +1* the robot remains in the same place is important. This is the reason why we measure the distance between the new signature and the one contained in the node *n*.

For the robot to change the place, the difference between the current signature and the signature of the node must be greater than the threshold $\tau = 0.2$.

In case the measurement of the similarity is above this threshold, the robot has to compare the distance between the current signature and all the signatures of the nodes in the map in order to extract the minimum distance. The latter provides information about the room closest to the current signature. If this distance is less than the threshold of the loop closing of the specified room, the robot must infer that this room is already visited. On the other hand, when the situation is reversed, it creates a new node that includes the new signature.

The detection of the loop closing provides information about places already visited by the robot allowing for a more robust creation of the map. When the robot encounters a visited node, a link is made between this node and the last node of the map to make it more consistent.

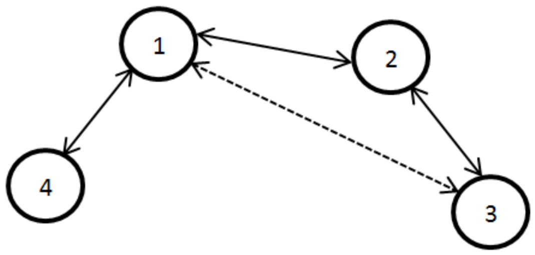


Figure 10.5: After creating the third node of the map, the robot encounters signatures that belong to a node already visited (i.e., the first). A link is set between these nodes to indicate the possibility of the passage.

Algorithm 6 Topological map building

```
Input Signature $S_t$, Current Signature $S_{t+1}$, Map $M_t$
Output $M_{t+1}$
CurrentNode$\leftarrow n_i$;
if (distance($S_t$,$S_{t+1}$)$\prec \tau$ )
    Save $S_{t+1}$ in $n_i$;
    $M_{t+1} \leftarrow M_t$;
else
    Search $n^* = arg \min_{n_k \in M_t}$ (distance($S_t$,$S_{t+1}$))
     if distance ($S^*$,$S_{t+1}$)$\preceq \tau_{loop}$
        Add connexion $n_i \mapsto n^*$;
        CurrentNode $\leftarrow n^*$;
     else
        Create a new node n;
        Save $S_{t+1}$ in n;
        CurrentNode$\leftarrow$n;
        $M_{t+1} \leftarrow M_t$+CurrentNode;
     endif
endif
Return $M_{t+1}$, CurrentNode
```

10.4 Experimental results

10.4.1 COLD dataset

The acronym COLD stands for COsy Localization Database. Pronobis and Caputo[184] made this database to represent a new collection of image sequences and provides a flexible testing environment that is mainly vision-based. It also aims to work on mobile platforms in real-world environments. The COLD database consists of three independently collected sub-datasets gathered over three distinct indoor laboratory environments, which contain spaces of common functionality, and are located in different European cities.

In this chapter, we used the COLD-Freiburg dataset that was acquired at the Autonomous Intelligent Systems Laboratory at the University of Freiburg in Germany (Figure10.6). For the data acquisition, COLD was extracted from omnidirectional and perspective cameras

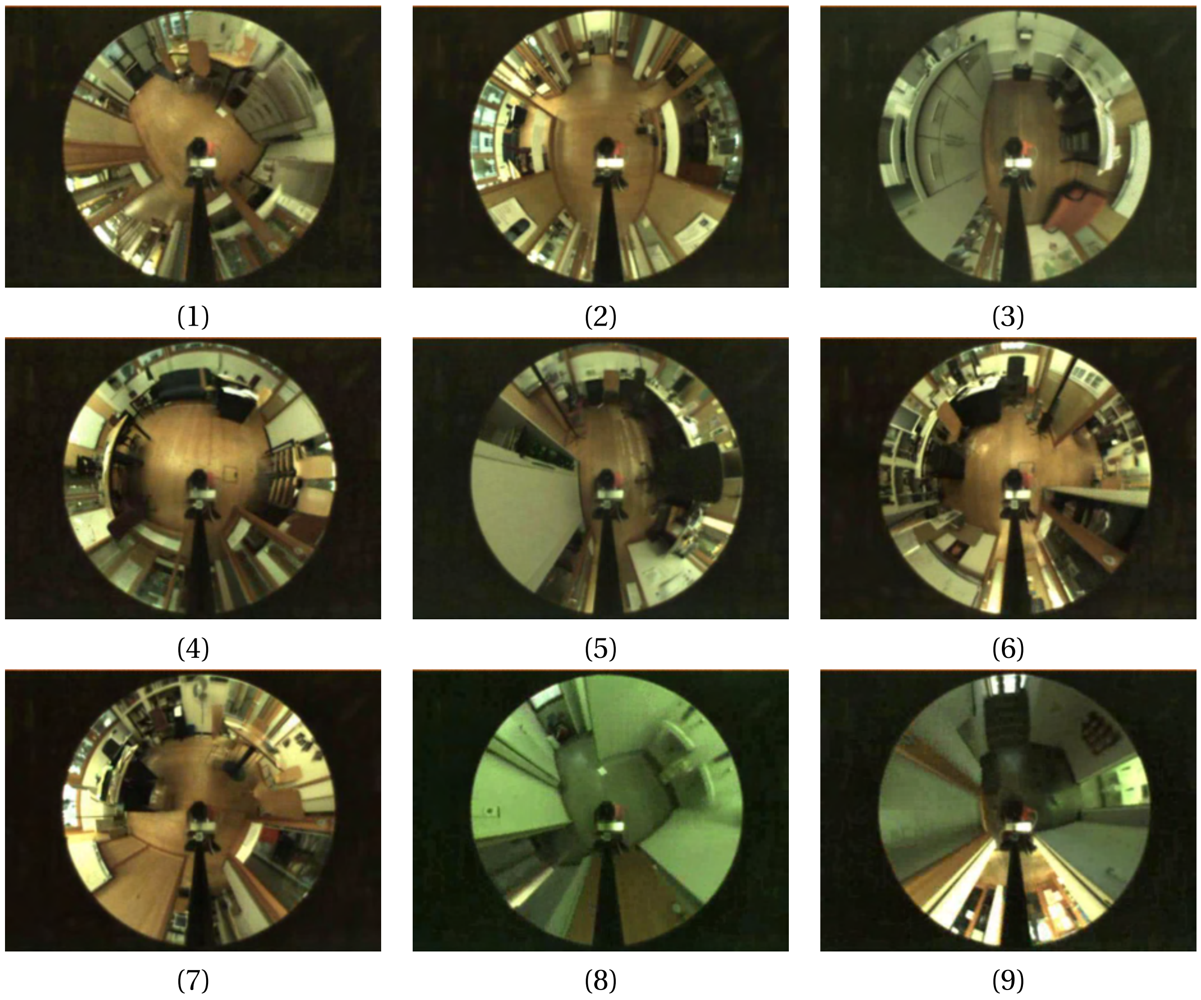


Figure 10.6: The sample frames extracted from video sequences of COLD-Freiburg dataset.

mounted together on a socket, which was moved from one lab to another. The socket was mounted on the robot platform available at each lab, and each robot was driven manually across several rooms for the data acquisition.

10.4.2 Topological mapping

We have carried out several experiments to test our approach of the topological navigation of a totally-independent mobile robot in an unknown environment, and we have validated it by video sequences issued from COLD-Freiburg dataset of omnidirectional images taken in an indoor environment (i.e., laboratory). Figure 10.7 shows our results of the map building process (i.e., topological graph).

The graph consists of nine nodes, each one representing and keeping different signatures of the laboratory rooms. We can find out that the node of the corridor stores a high number of signatures (i.e., 1118 frames). This is due to the robot needing to visit the corridor every now and then in order to move from one room to another in a given environment. Table 10.1 summarizes the number of signatures emanating from each room.

Laboratory rooms	Index	Frames
Printer area	1	302
Corridor	2	1118
Kitchen	3	193
Large office	4	211
Two-persons office 1	5	90
Two-persons office 2	6	207
One-person office	7	58
Bath room	8	122
Stairs area	9	571

Table 10.1: Number of signatures saved in each node of the map-graph.

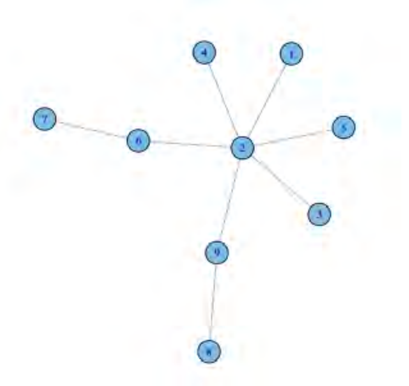


Figure 10.7: Topological map building.

10.4.3 Detection of loop closing

In this experiment, during the creation of the topological map, the robot tests the similarity between the signature of the frame *t=1* and all the other signatures of the nodes.

The minimum distance 0.0921 resulting from the test represents the distance between the signature of the frame *t=408* and frame *t=1*, which belongs to the category corridor. In order to ensure that the current signature *t=408* pertains to the corridor category, the robot must ascertain that the distance is less than the threshold of the loop closing $\tau_{loop} = 0.1809$ associated with this category (see Table 10.2).

The robot also encounters other loop closings at t=851, t=1241, t=1551 and t=2037. So, for it to navigate through the whole lab, it must necessarily visit the corridor to reach the other places.

Laboratory rooms	Index	τ_{loop}
Printer area	1	0.2484
Corridor	2	0.1809
Kitchen	3	0.4566
Large office	4	0.5258
Two-persons office 1	5	0.4189
Two-persons office 2	6	0.5966
One-person office	7	0.3828
Bath room	8	0.2332
Stairs area	9	0.0940

Table 10.2: Loop closing thresholds for each category of nodes.

frame t=1

frame t=2

frame t=3

frame t=4

frame t=5

frame t=6

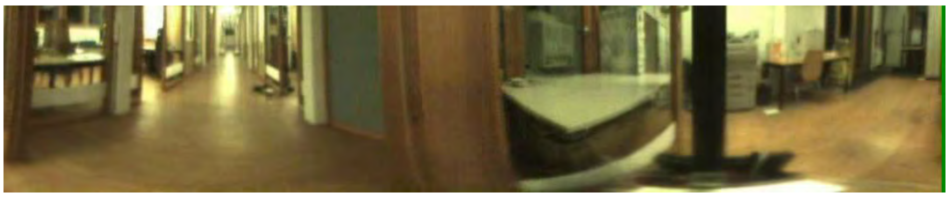

frame t=408

Figure 10.8: The sample frames extracted from video sequences. The loop closing is detected in the frame t=408.

10.5 Conclusion

The approach presented in this chapter introduced a new method of constructing topological maps in unfamiliar environments using omnidirectional vision only. Its originality stems out of the quality of the environment representation, which is based on compact and fast global features. The results obtained here show the robustness of the loop closing detection, which uses a statistical formula in order to build a coherent topological map. Nevertheless, our method has some limitations when, for example, the robot acquires images in a random manner. Therefore, to overcome such limitations, the robot must have a good exploration strategy and a well-planned procedure to acquire omnidirectional images. Also, we will propose a new place recognition-based approach using deep learning architectures for both feature extraction and classification.

Chapter 11

End to End Learning to Navigate in Indoor Envrionment

"Study the past if you would define the future."

Confucius

Contents

RECENTLY, place recognition have become an important issue for autonomous mobile robots. It is mainly used for robot navigation in indoor environments. In this chapter, we propose new topological navigation based on place recognition. To let the robot has the ability to explore and navigate in its environment, it must be able to identify the places of the environment of varying duration. For this purpose, we use Convolutional Long Short-Term Memory (C-LSTM) architecture to learn places during mobile robot navigation. The C-LSTM involves Convolution Neural Network (CNN) layers to extract features from the input data combined with Long Short-Term Memory (LSTM) to consider the information of the previous frames and to learn the temporal dependencies of the robot movement. For feature extraction, we used the pre-trained CNN on the scene-centric dataset called Places, then the extracted features are used as LSTM inputs. After performing the place recognition, the robot creates and updates its topological map in order to act in its environment. In summary, the main contributions are:

- we propose a new place recognition-based approach for topological navigation;
- we use C-LSTM architecture to learn places;
- we use the pre-trained CNN on Places dataset to extract features and LSTM to learn temporal dependencies of the data;
- we build a topological map based on place recognition.

11.1 Introduction

Mobile robots are intelligent machines that are designed to solve hard tasks in various circumstances. Since the creation of the first intelligent machine, robots have served humans in various aspects. Nowadays, robots are not just replacing humans in industrial assembly lines but they become a part of our daily life. They are present in the shopping, tour guide, entertainment, housework, waiter service, education, and so on. In all these domains, the robot should be able to provide an efficient representation of the environment, which can be used as a common understanding among all involved subjects in the scenario. Also, it can support the robots to fulfill assigned missions.

Generally, metric and topological maps are the two fundamental types of environment representations. Metric maps describe the surrounding environment in a measurable and precise way. In such maps, the environment is represented by a set of objects with associated positions in a metric space, usually in two dimensions. This space is mostly the one in which the position of the robot estimated by the proprioceptive data is expressed. Using a metric model of the sensors, perceptions allow detecting these objects and estimating their position relative to the robot. The position of these objects in the environment is then calculated using the estimated robot position. In some locations, these objects correspond to the obstacles that the robot may encounter in its environment. The environmental map then corresponds directly to the free space, in which the robot can move. Although metric maps encounter some difficulties in obtaining a metric model of the sensors. In order to efficiently represent the environment, topological maps are widely used for several vision-based applications since they contain sufficient information excluding overly detailed metrics. They can be viewed as abstract representations that describe the links between elements in the environment without using an absolute reference frame. The nodes of the graph correspond to places. While the edges linking these nodes mark the transition from one place to another and memorize in general, the manner of making this transition. Topological maps are useful for path-planning and localization.

Topological mapping and scene recognition are the most important issues to model environments with sparse information. Humans describe where they are by using unique labels of the places such as "office", "the first corridor", "my workspace", and so on. According to the humans' psychology [156], region-based topological structures are mostly used by humans when such information is learned or recognized. Consequently, humans can understand their surroundings by mostly topological meanings. Thus, topological modeling can serve also for mobile robot navigation through topological nodes which represent the specific places of the environment.

Recently, typical deep learning methods including CNNs and RNNs have yielded state-of-the-art results for a wide variety of tasks in the field of audio processing, natural language processing, and computer vision such as object classification, image caption generation as well as place classification. CNNs were used in several times for place recognition tasks by applying a convolutional layer to extract features from local patches of the data. CNNs often used a pooling operation to pool values of features over neighboring patches. Then the extracted features can serve as the input of the next layer in the neural network, possibly other convolution layers or a classifier. Whereas, RNNs are specific models for processing sequential data that are frequently of a varied length such as text or sound. RNNs consist of high dimensional hidden states that work as the memory of the network in which the state of the hidden layer at a time t is conditioned on its previous state. This structure enables the RNNs to store, remember, and process past complex signals for long time periods. LSTM is a special type of RNNs that use a gating concept for better modeling of long-term dependencies in the data. This type has been used successfully for machine translation and speech recognition. In the case the data is a sequence of frames and of a varied length, a model can combine both convolutional and recurrent layers. The convolutional layers can be used to extract features from data patches of sequence frames and can serve as the input of recurrent layers modeling the temporal relations of the frames. The main advantage of this method is that using a pooling technique after the convolution layer can shorten the input sequence to the recurrent layer to model temporal dependencies over a small number of frames.

In this chapter, we suggest a new place recognition-based navigation approach using topological maps. The approach enables mobile robots to recognize the visited indoor places based only on images which are sequentially captured. Then integrates or updates automatically the nodes into the incremental topological map. To perform place recognition, we used Convolutional Long Short-Term Memory (C-LSTM) architecture to learn places during mobile robot navigation. The C-LSTM consists of CNN layers extracting features from the input data and LSTM considering the temporal dependencies of the robot movement. In the feature extraction, we used the pre-trained CNN on "Places" dataset, then those features are used as LSTM input. After place recognition step, the robot creates and updates its topological map in order to act in its environment. Figure 11.1 depicts the major steps of our topological mapping approach:

- we use the pre-trained CNN on Places dataset to extract features;
- we use LSTM layer to learn temporal dependencies of the data;
- we build a topological map based on place recognition.

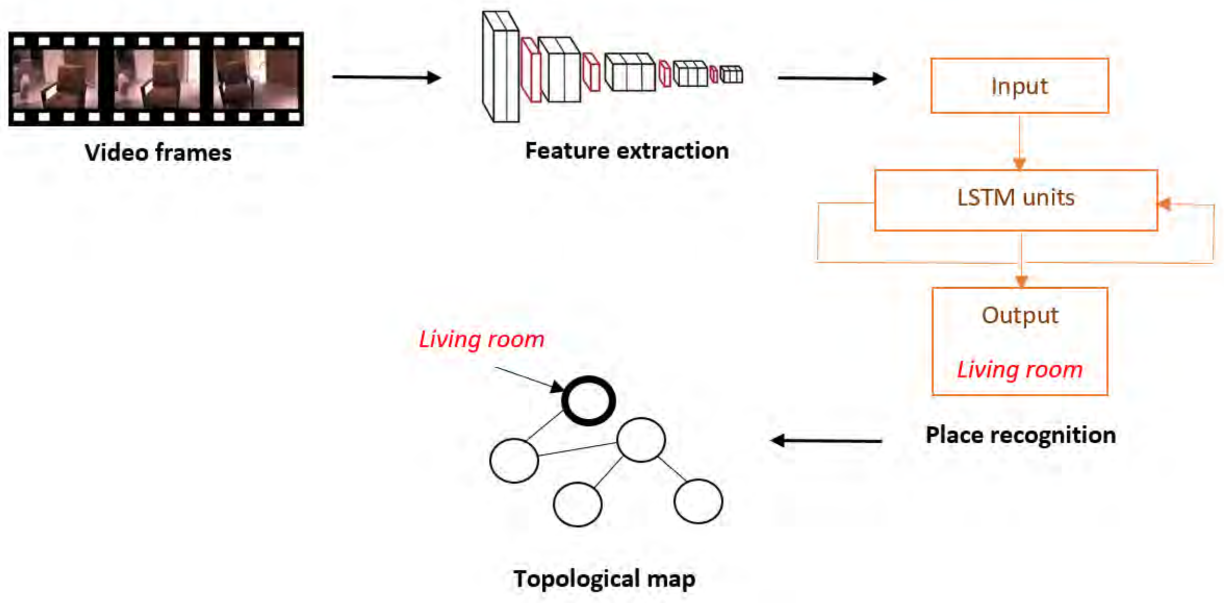


Figure 11.1: Overview of our topological mapping approach.

11.2 Our topological map

Topological maps are relatively different from metric ones [159], as place definitions and their relative positions are recorded respectively in the map nodes and edges. Usually, nodes memorize places without the need to resort to metric models of the sensors, while edges store relative positions of the nodes, ranging from simple adjacency information to precise relative metric positions.

Here, we propose to construct a topological map by representing the indoor environment as graph form. As a result, each node corresponds to a certain partially enclosed area within the environment (e.g. a room, a kitchen, a workspace) which is connected to neighboring nodes. The nodes are designed by our place recognition approach C-LSTM. Whereas, edges represent the connectivity between places and correspond to transitions from one place to another. These connections can be viewed as doors, corridors, or just a simple transition from the actual place to its close neighbor. This topological map representation resembles the way humans perceive their environments during navigating and is likewise convenient from a robot's navigation perspective.

11.3 Convolutional Long Short-Term Memory (C-LSTM)

Instead of using single images, we perform place recognition as a video classification problem. Since the videos are considered as sequential data, we add an LSTM layer on top of CNN, thus extracting temporal dependencies from different sequence frames. Our C-LSTM model is primarily based on using CNN since they are excellent at extracting pertinent features from images. Training CNN for image representation requires thousands of images and also high processing power such as Graphics Processing Unit (GPU) for the weight adjustment of the model. Getting such a model using this strategy is an expensive process, however, we used transfer learning to handle these problems.

Algorithm 7 illustrates the three main steps of our C-LSTM model. In the pre-processing step, we convert all the video sequences to frames then we split the data into training and validation sets. After that, we reshape all the data into size 224×224×3 to be compliant to the standard input of the VGG16. In the feature extraction step, we used the pre-trained VGG16 model that was trained on the Places365 dataset (Section 7.4.4). First, we initialize only the convolution blocks of the VGG16 model [222]: Conv block 1 with 64 output filters, Conv block 2 with 128 output filters, Conv block 3 with 256 output filters, Conv block 4 with 512 output filters, and Conv block 5 with 512 output filters. Second, we extract features layer by layer, from simple representation to complex concepts. Finally, the extracted features are stored in order to be used as LSTM inputs. After performing feature extraction, we add an LSTM layer to the model in order to learn the temporal relationship and structure from the extracted features. The final result was obtained from a softmax layer. Our C-LSTM model learns end to end features from video sequences and benefits from: 1) the efficiency of CNNs in implicit feature extraction and 2) the ability of RNNs in modeling sequential data.

Algorithm 7 Convolutional Long Short-Term Memory (C-LSTM)

```
1) Data pre-processing
    Convert videos to frames
    Split the frames into training and validation data
    Resize the frames following the standard VGG16 input (224×224×3)
2) CNN: feature extraction
    Use pre-trained CNN on Places365
    Load the convolution blocks of the VGG16 model
    Extract and save the CNN features
3) LSTM: fine-tuning
    Initialize the LSTM by CNN features
    Train LSTM on the extracted features
    Fine-tune the C-LSTM network
```

11.4 Topological mapping

Since the indoor environments are composed of different rooms and corridors which are connected by different transitions, they can be represented into a topological map. This abstract representation contains many nodes as the number of rooms and corridors and several edges which mark all the possible transitions in the environment.

When the robot begins the exploration of a given environment, its strategy is to carefully record several sequences of frames in every discovered room. Then, it should learn the environment model using our C-LSTM architecture. After creating the place recognition model, the robot creates and updates its topological map based only on the captured frames in every time t.

Initially, the environment map M_t is empty, the robot captures the first frame F_t then predict the label of this place using C-LSTM model. Since the map is empty, the initial recognized frame is stored by default in the node $n_{\mathrm{L}abel}$ which contains the index of the place label. Then, the robot updates its map M_t. During the robot's navigation, new frames are created allowing it to define new places and get situated. When the robot remains in the same place, the label of the previous frame and the current one are similar. In this case, it only stores this frame in the previous node and proceeds its navigation. When the algorithm predicts another label, the robot encounters a new place and updates its map M_{t+1} by adding a new node that bears the index of the current label. To respect the topological structure, the robot adds a new connexion (i.e., edge) between the previous node and the current one, thus indicating the transition and the neighborhood between these two places.

Also, Algorithm 8 manages the loop closing problem which determines whether the currently observed node is previously visited, or a new place of the environment being explored. Before assigning a new node to the map M_{t+1}, the robot compares the current label with all the indices of the stored nodes in the map in order to prevent redundancy nodes in the final map.

Algorithm 8 Topological mapping

```
**Input**: C-LSTM model, Map $M_t = \emptyset$, Input Frame $F_t$,
**Output**: $M_{t+1}$
Label ← Predict $F_t$ using C-LSTM model
**if** ($M_t == \emptyset$)
  Create a new node $n_{Label}$;
  CurrentNode← $n_{Label}$
  Save $F_t$ in $n_{Label}$
  $M_t \leftarrow M_t$+CurrentNode;
**else**
  **if** ( $M_t.n.Label_i$== Label s.t. $\exists$ i $\in$ {$M_t.n.Label$})
    Save $F_{t+1}$ in $n_{Label}$;
    $M_{t+1} \leftarrow M_t$;
  **else**
    Create a new node $n_{Label}$;
    Save $F_{t+1}$ in $n_{Label}$;
    Add connexion ($n_{Label}, M_t.n$);
    CurrentNode← $n_{Label}$
   $M_{t+1} \leftarrow M_t$+CurrentNode;
  **endif**
**endif**
**Return** $M_{t+1}$, CurrentNode
```

11.5 Experimental results

In this section, we first introduce the publicly available robot vision datasets, the experimental setup, followed by the experiments presented in two parts: C-LSTM scene recognition, and topological map building.

11.5.1 Datasets

To evaluate our method, we use four publicly available datasets for place recognition and robot localization: NYU V1, NYU V2, Visual Place Categorization (VPC), and COsy Localization Database (COLD).

NYU V1

The NYU V1 dataset [218] was previously introduced in Section 8.5.1. It was captured from a range of indoor environments such as residential apartments, workplace and university campus settings. NYU V1 contains 2347 pairs of images spread over 64 different indoor environments which are grouped into seven categories, including a bathroom, a bedroom, a bookstore, a cafe, a kitchen, a living room, and an office. In this experiments, we used only the RGB images since our C-LSTM approach is designed particularly for RGB data.

NYU V2

NYU V2 dataset [219] was introduced in Section 8.5.1 to evaluate our RGBD scene classification. Here, we used only RGB images. The original 27 categories are reorganized into 10 scene categories, including the 9 most common categories and another category for images in the remaining categories. The 9 most common categories include a bathroom, a bedroom, a bookstore, a classroom, a dining room, a home office, a kitchen, a living room, and an office kitchen.

Visual Place Categorization (VPC)

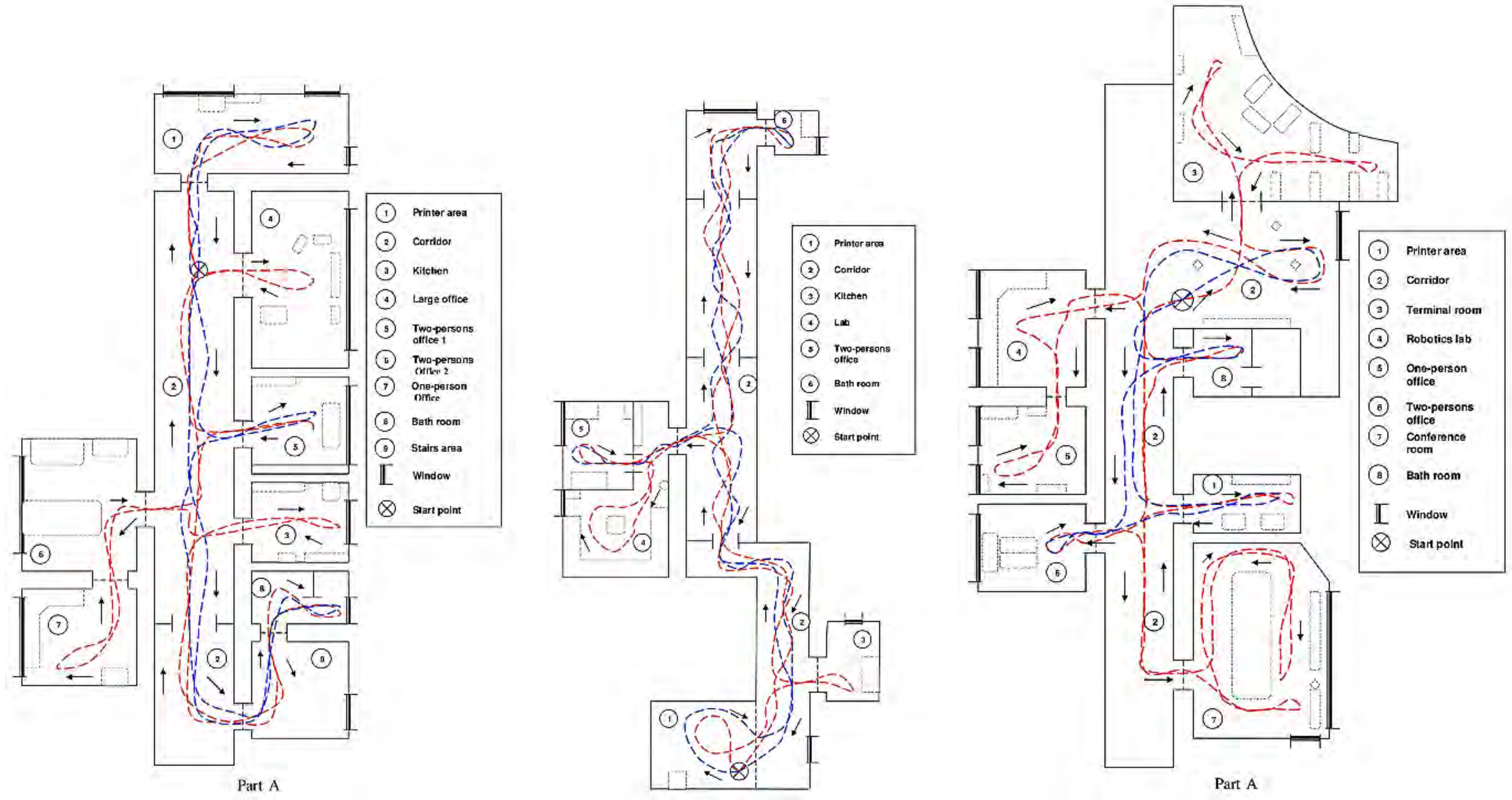
Visual Place Categorization (VPC) dataset consists of videos captured autonomously using a rolling tripod plus an HD camcorder (JVC GR-HD1) to mimic a robot. VPC collects videos from six home environments. In Section 7.4.1, we provided some image samples from this dataset.

COsy Localization Database (COLD)

The acronym COLD stands for COsy Localization Database [184]. It represents a new collection of image sequences and provides a flexible testing environment that is mainly vision-based. The COLD database consists of three independently collected sub-datasets gathered over three distinct indoor laboratory environments (i.e., Freiburg, Ljubljana, and Saarbrücken) acquired at three different indoor laboratory environments located in three different European cities: the Autonomous Intelligent Systems Laboratory at the University of Freiburg Germany, the Visual Cognitive Systems Laboratory at the University of Ljubljana Slovenia, and the Language Technology Laboratory at the German Research Center for Artificial Intelligence in Saarbrücken Germany. The data were recorded using three different mobile robot platforms and the same camera setup, under various weather and illumination conditions (i.e., during cloudy weather, sunny weather, and at night) over several days. As shown in Figure 11.2, in each laboratory, the robot roughly followed two different paths: standard (i.e., consisting of rooms that are most likely to be found in most typical office environments) and extended (i.e., additionally containing rooms that were specific to this environment or its part).

11.5.2 Experimental setup

The datasets are divided by following machine learning three splits protocol in training, validation, and testing of 60%, 20%, and 20%, respectively. Then, we reshape the images into size 224×224 in order to be compliant to the standard input of CNN. In the feature extraction, we load the VGG16 model pre-trained on Places365 dataset except the top model in order to extract and save the features which represent the inputs of the next LSTM layer. The LSTM layer has 128 internal cells for each time step. In addition, we use dropout to prevent overfitting. After that, we fine-tune the C-LSTM, using the stochastic gradient descent (SGD) with 0.9 momentum and 1e-6 weight decay. The learning rate is set to 0.001 and decays by a factor of 0.9. These parameters were selected empirically, after a set of experiments. This model was implemented in python and trained using Keras framework supported by TensorFlow in the backend. Table 11.1 summarizes the general characteristics used in our C-LSTM experiments.



Freiburg map Ljubljana map Saarbrücken map

Figure 11.2: COLD maps for every laboratory. Blue color represents the standard path, while red color represents the extended path.

Characteristics	**Values**
LSTM units	128
Optimizer	Stochastic Gradient Descent (SGD)
Learning rate	0.001
Learning rate decay	1e-6
Momentum	0.9
Loss	categorical cross entropy
Batch size	32
Dropout	0.5
Epochs	1000

Table 11.1: Keras characteristics used in our C-LSTM experiments.

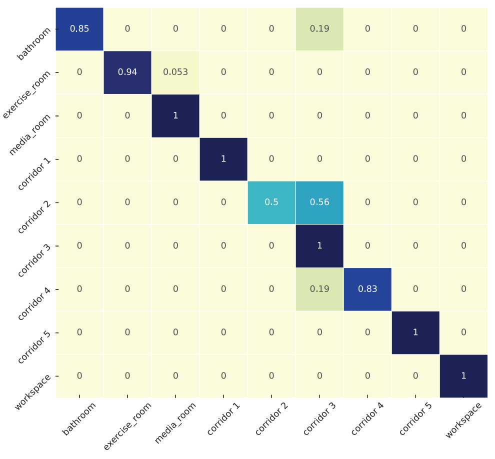
11.5.3 C-LSTM scene recognition results

CNN is a dominant deep learning architecture for both the representation and the classification of images. In our case, we used video data of a mobile robot's navigation on different indoor environments. We represent each individual frame by CNN features, followed by finding the temporal information between them using the LSTM layer with 128 units. For comparison, we firstly perform the experiments using the pre-trained VGG16 model on Places dataset, which contains five convolutional layers with max-pooling after each. We remove the last fully-connected layer whose layer's outputs are the 365 class scores for Places365. Then, we extract the CNN features for all the images. After that, we train the CNN model using the similar parameters of Table 11.1. To perform the classification, we add two fully-connected layers, two dropout layers, and finally one softmax layer to predict the probabilities. Tables 11.2-11.7 show the comparison between the CNN implementation for feature extraction and classification as well as the C-LSTM one for CNN feature extraction and LSTM classification. All the experiments assure that our C-LSTM performs better than CNN thanks to the temporal dependencies between the sequential data which are presented as robot motion videos.

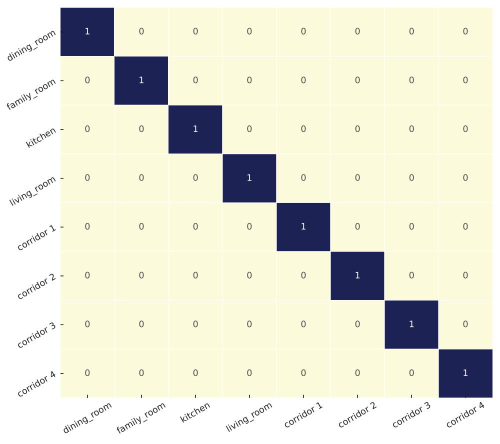
	home 1	**home 2**	**home 3**	**home 4**	**home 5**	**home 6**
CNN Accuracy	84.37%	96.87%	86.32%	73.43%	84.89%	94.79%
CNN Precision	90%	97%	90 %	82%	90 %	96 %
CNN Recall	84%	97 %	86 %	73%	85%	95%
C-LSTM Accuracy	**90%**	**100%**	**89.84%**	**79.29%**	**88.54%**	**97.91%**
C-LSTM Precision	95 %	100%	94 %	84%	90 %	98 %
C-LSTM Recall	90%	100%	90%	79%	89 %	98 %

Table 11.2: Recognition rates for the VPC dataset.

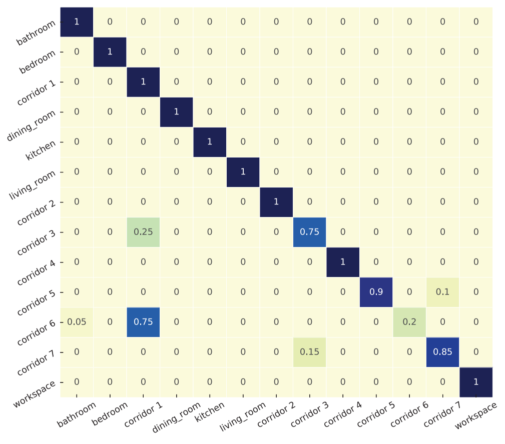
The resulting performance on VPC dataset is given in Figure 11.3 and Table 11.2. The confusion matrixes of all the homes show that corridor classes are generally confused. This outcome is evident since corridors do not have a specific property such as the rooms of the environments. They are very simple often walls with some differences. On the other hand, the rooms contain significant structures such as furniture, decorations, colors, and textures, which represent high-level characteristics. Homes 2 and 6 are perfectly classified with the accuracy 100% and 97.91% respectively as they contain the least number of corridors compared to the other.



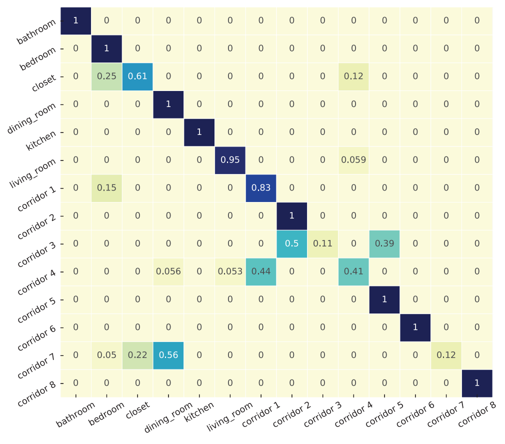
home 1



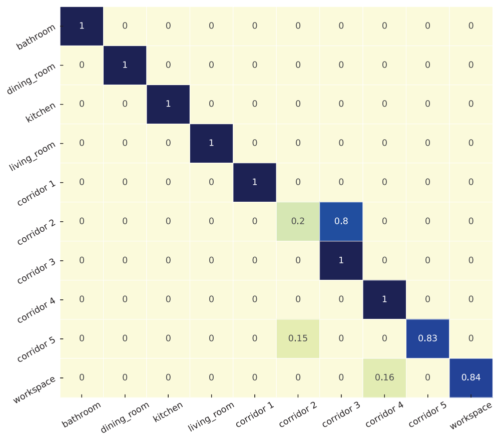
home 2



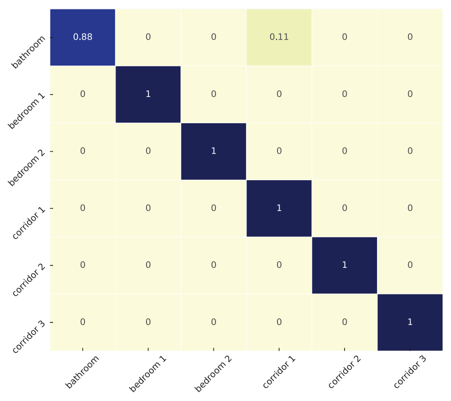
home 3



home 4



home 5



home 6

Figure 11.3: Confusion matrixes of our C-LSTM approach on VPC dataset.

As shown in Figure 11.4 and Table 11.3, the performance of C-LSTM approach on NYU V1 achieves the value 94.87%. The classes which are misclassified are a bookstore with a cafe and a kitchen with an office, which share some common properties. Also, NYU V2 reaches the value 90.72% that represents considerable improvements in the state-of-the-art. Table 11.4 shows comparisons with the approaches [54, 68, 74, 102, 228, 237, 261]. Gupta *et al.* [68] provided a scene classification approach which used the semantic segmentation maps, spatial pyramid, and SVM. Hayat *et al.* [74] suggested spatial layout and scale invariant convolutional activations. Khan *et al.* [102] exploited rich mid-level convolutional features which are extracted from uniformly and densely image patches using deep CNNs. These mid-level features are encoded in terms of their association with the codebooks of Scene Representative Patches (SRPs).

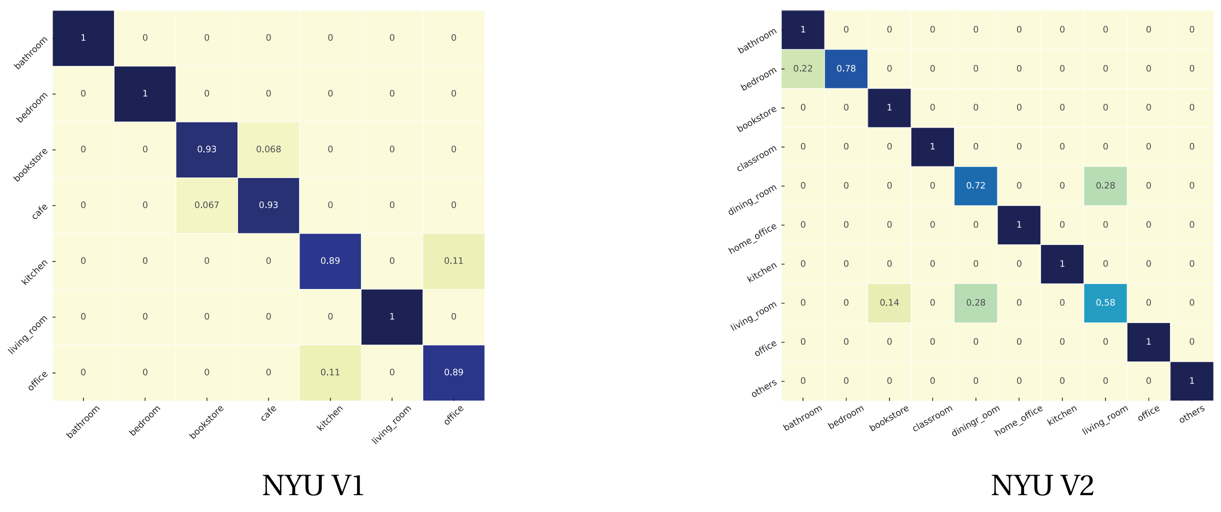


Figure 11.4: Confusion matrixes of our C-LSTM approach on NYU V1 and V2 datasets.

	NYU V1	**NYU V2**
CNN Accuracy	91.82%	86.29%
CNN Precision	93 %	87%
CNN Recall	92 %	86%
C-LSTM Accuracy	**94.87%**	**90.72%**
C-LSTM Precision	95%	91 %
C-LSTM Recall	95 %	91 %

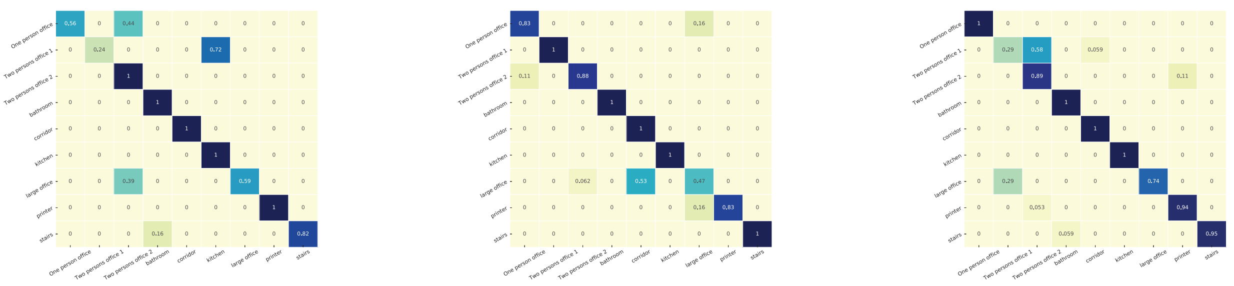
Table 11.3: Recognition rates for NYU V1 and V2 datasets.

Tao *et al.* [237] proposed a Rank Preserving Sparse Learning (RPSL) which takes into consideration four aspects: the first aspect maintained the rank order information of the within-class samples in a local patch while it ignored the rank order information of the between-class samples, the second aspect maximized the margin for the between-class samples on a local patch, the third aspect introduces the L1-norm penalty to obtain the sparse representation, and the final aspect models the classification error minimization that uses the least squares error minimization. Feng *et al.* [54] suggested Discriminative Locality Alignment Network (DLANet) that adopted the PCANet structure which learned the convolutional filter bank through PCA in order to learn the local features. Wang *et al.* [261] investigated a framework that allows greater spatial flexibility, in which the Fisher Vector (FV) encoded distribution of local CNN features. These features are extracted from an augmented pixel-wise representation comprising multiple modalities of RGB, HHA, and surface normals to capture more information about the geometry. Song *et al.* [228] focused on the bottom layers and proposed an alternative strategy to learn depth features combining local weakly supervised training from patches followed by global fine tuning with images. In this chapter, we experiment our C-LSTM approach using only the RGB images of NYU V1 and V2 datasets. The results are very remarkable and show large improvements compared to the state-of-the-art.

Approaches	**NYU V1**	**NYU V2**
Gupta *et al.* [68]	- -	58%
RGBD-LLC [237]	78.1%	- -
DLANet [54]	80.33 %	- -
S^2ICA [74]	81.2%	- -
DUCA [102]	80.6%	- -
Combined FV and Full [261]	- -	53.5 %
RGB-D-CNN [228]	- -	53.4%
Our C-LSTM	**94.87** %	**90.72%**

Table 11.4: Comparison of state-of-the-art and our C-LSTM approach on NYU V1 and V2 datasets.

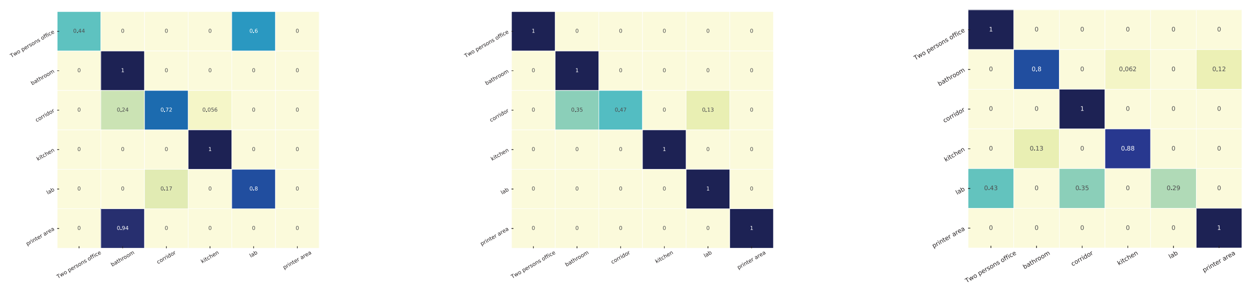
Moreover, we have evaluated the performance of our C-LSTM approach using omnidirectional images. Figure 11.5 illustrates the confusion matrixes of our C-LSTM approach on Freiburg lab extended path under different illumination: cloudy, night, and sunny. The results confused between one person office, two person's office 2, and large office which are from the same category office. Also, we can notice that the night illumination provides more accurate results than those in cloudy and sunny experiments.



Cloudy Night Sunny

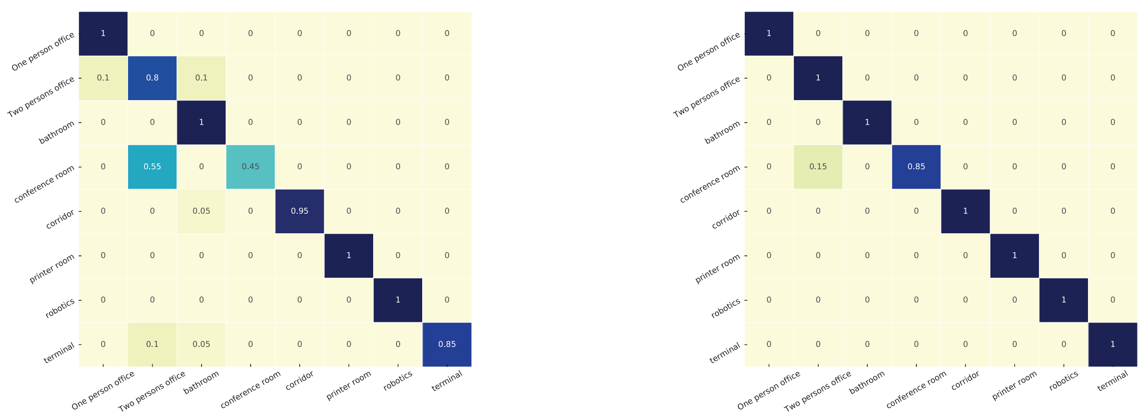
Figure 11.5: Confusion matrixes of our C-LSTM approach on Freiburg lab extended path under different illumination conditions: cloudy, night, and sunny.

Figure 11.6 depicts the confusion matrixes of our C-LSTM approach on Ljubljana lab extended path. In the cloudy experiment, a printer area class is totally confused with the bathroom one. Contrary to night and sunny experiments, printer area was well-classified thanks to the illumination conditions which differentiates between these two classes. Also, we observed that the majority of the classification errors of our C-LSTM occurred during the transition between different places, specifically adjacent places with the possibility of overlapping scenes such as the corridor with the other rooms. Similarly, Saarbrücken lab encounters perfect classification with the night illumination in which only conference room class was confused with two persons office class (Figure 11.7).



Cloudy Night Sunny

Figure 11.6: Confusion matrixes of our C-LSTM approach on Ljubljana lab extended path under different illumination conditions: cloudy, night, and sunny.



Cloudy Night

Figure 11.7: Confusion matrixes of our C-LSTM approach on Saarbrücken lab extended path under different illumination conditions: cloudy, and night.

	Freiburg lab			Ljubljana lab			Saarbrücken lab	
	cloudy	**night**	**sunny**	**cloudy**	**night**	**sunny**	**cloudy**	**night**
CNN Accuracy	77.50%	86.25%	81.25%	58%	81.25%	78.12%	84.37%	88.75%
CNN Precision	88%	89%	87%	61%	88%	77%	83%	91%
CNN Recall	78%	86%	81%	58%	81%	78%	84%	89%
C-LSTM Accuracy	**80.62%**	**88.75%**	**86.87%**	**67%**	**91.66%**	**82.29%**	**88.12%**	**98.12%**
C-LSTM Precision	89%	90%	87%	64%	94%	86%	91%	98%
C-LSTM Recall	81%	89%	87%	67%	92%	82%	88%	98 %

Table 11.5: Recognition rates for the COLD dataset extended path under different illumination conditions.

For sake of completeness, we also report Tables 11.5- 11.7 associated with our results on COLD dataset under different illumination conditions. Table 11.5 represents recognition rates for the extended path, whereas Table 11.6 is devoted to the standard one. The results are very encouraging especially those of the standard path which consists of four or five classes per laboratory. The recognition rates vary from 96.87% to the highest value 100% which can be used effectively for mobile robot navigation in a similar environment. Moreover, Table 11.7 shows the recognition rate results of the COLD dataset with varying learning and recognition scenarios. We used the cloudy data as learning, then we tested the model with night and sunny data. The results clearly demonstrate significant robustness to lighting variations and assert that our model can be trained offline and applied on robotic platforms navigating under different illumination conditions.

	Freiburg lab			**Ljubljana lab**			**Saarbrücken lab**	
	cloudy	**night**	**sunny**	**cloudy**	**night**	**sunny**	**cloudy**	**night**
CNN Accuracy	95.83%	95.83%	94.79%	98.43%	93.75%	92.18 %	92.18%	98.43 %
CNN Precision	96%	97%	95%	99%	95%	94%	93%	95%
CNN Recall	96%	96%	95%	98%	94%	92%	92%	94%
C-LSTM Accuracy	**98.95%**	**100%**	**100%**	**100%**	**98.43%**	**96.87%**	**96.87%**	**100%**
C-LSTM Precision	99%	100%	100%	100%	99%	97%	97%	100%
C-LSTM Recall	99%	100%	100%	100%	98%	97%	97%	100%

Table 11.6: Recognition rates for the COLD dataset standard path under different illumination conditions.

	Freiburg lab		**Ljubljana lab**		**Saarbrücken lab**
	night	**sunny**	**night**	**sunny**	**night**
CNN Accuracy	64.58%	70.83%	75%	89.06%	93.75%
CNN Precision	67%	83 %	62%	91%	95%
CNN Recall	65%	71%	75%	89%	94%
C-LSTM Accuracy	**69.79%**	**80.20%**	**78.12%**	**93.75%**	**100%**
C-LSTM Precision	70%	89%	66%	94 %	100%
C-LSTM Recall	70%	80%	78 %	94 %	100%

Table 11.7: Recognition rates for COLD dataset with varying learning and recognition scenarios.

The comparison of our C-LSTM approach and the state-of-the-art is given in Table 11.8. Lin *et al.* [133] proposed a scene recognition technique related to the human perception for the scene change detection as well as a topological map construction using omnidirectional image sequences. They provided Extended Hull Census Transform (Extended-HCT) semantic scene descriptor which is based on the SURF features. Erkent and Bozma [51] provided Bubble Space (BuS) based representation of "places" (i.e., nodes) in topological maps. BuS simultaneously provided for detailed (i.e., bubble surfaces) and holistic (i.e., bubble descriptors) representation of places. It is based on bubble memory where visual feature values and their local S2-metric relations from the robot's viewpoint are simultaneously encoded on a deformable spherical surface. The results demonstrate that our model achieves comparable performance with previous works, showing the advantage of combining CNN features with the temporal ability of LSTM.

11.5.4 Topological mapping

A mobile robot first learns each place at each laboratory in one illumination condition using the C-LSTM approach. Once it completes learning, it predicts every frame during its navigation, thus building a topological map of the environment. Throughout its deplacement,

Datasets	**BuS approach** [51]	**Lin *et al.*** [133]	**Our approach**
Freiburg extended cloudy	58.9%	- -	**80.62%**
Freiburg extended night	72.5%	- -	**88.75%**
Freiburg extended sunny	63.2%	- -	**86.87%**
Freiburg standard cloudy	90%	67.16%	**98.95%**
Freiburg standard night	77.1 %	73.30%	**100%**
Freiburg standard sunny	89.5 %	71.04%	**100%**
Ljubljana extended cloudy	81.3 %	- -	67%
Ljubljana extended night	88.6 %	- -	**91.66%**
Ljubljana extended sunny	76.6 %	- -	**82.29%**
Ljubljana standard cloudy	82.2 %	- -	**100%**
Ljubljana standard night	85.6 %	- -	**98.43%**
Ljubljana standard sunny	90.1 %	- -	**96.87%**
Saarbrücken extended cloudy	72.3 %	- -	**88.12%**
Saarbrücken extended night	74 %	- -	**98.12%**
Saarbrücken standard cloudy	84.8 %	- -	**96.87%**
Saarbrücken standard night	84.5 %	- -	**100%**

Table 11.8: Comparison of state-of-the-art and our topological mapping approach on COLD dataset.

the robot may revisit some places which are previously stored in the map nodes. In such a situation, it should be able to update its map without duplicating the existing nodes. The experiments are performed on the video sequence of mobile robot navigation. A video is a combination of frames moving approximatively at 30 frames per second. However, several frames have much redundant information, whose processing is computationally expensive. To handle this problem, we jump five frames when processing a video for place recognition which does not affect the performance of the results. Table 11.9 provides the number of revisited places during each experiment and the loop closing detection on COLD, VPC, and NYU V1/V2 datasets. The results demonstrate that the robot can perfectly recognize the revisited places and by the way build a coherent topological map.

Figures 11.8-11.11 show the topological maps constructed from the COLD dataset images captured from the sequences Freiburg standard cloudy, Ljubljana standard night, Ljubljana standard sunny, and Saarbrücken standard cloudy. Black stars mark the robot movement in the laboratory, while the red ones are designed to show the misclassification of the current frame. We depict the confused frames for each sequence. Freiburg standard cloudy map (Figure 11.8) shows that only corridor frame is confused with printer area class. Also, in Figure 11.9, corridor frame is confused with bathroom class. These results are evident since the corridors frames contain a part of the printer area and bathroom. Similarly, in Ljubljana standard sunny map the robot misclassified two corridor frames with bathroom class, which are acquired in the transition between the bathroom and the corridor. As depicted in Figure 11.11, three frames are misclassified, two bathroom frames are confused with printer area and the corridor frame is confused with the bathroom. In general, the maps are very suitable since the end to end learning using our C-LSTM classficiation approach provided robust models for place recognition.

Datasets	Revisited places	Loop closing detection
Freiburg extended cloudy	20	10
Freiburg extended night	20	14
Freiburg extended sunny	20	13
Freiburg standard cloudy	14	14
Freiburg standard night	14	14
Freiburg standard sunny	14	14
Ljubljana extended cloudy	20	6
Ljubljana extended night	20	18
Ljubljana extended sunny	20	11
Ljubljana standard cloudy	14	14
Ljubljana standard night	14	13
Ljubljana standard sunny	14	12
Saarbrücken extended cloudy	20	15
Saarbrücken extended night	20	20
Saarbrücken standard cloudy	14	13
Saarbrücken standard night	14	14
home 1	40	37
home 2	40	40
home 3	40	37
home 4	40	32
home 5	40	37
home 6	40	39
NYU V1	90	86
NYU V2	90	82

Table 11.9: Revisited places and loop closing detection on COLD, VPC, and NYU V1/V2 datasets.

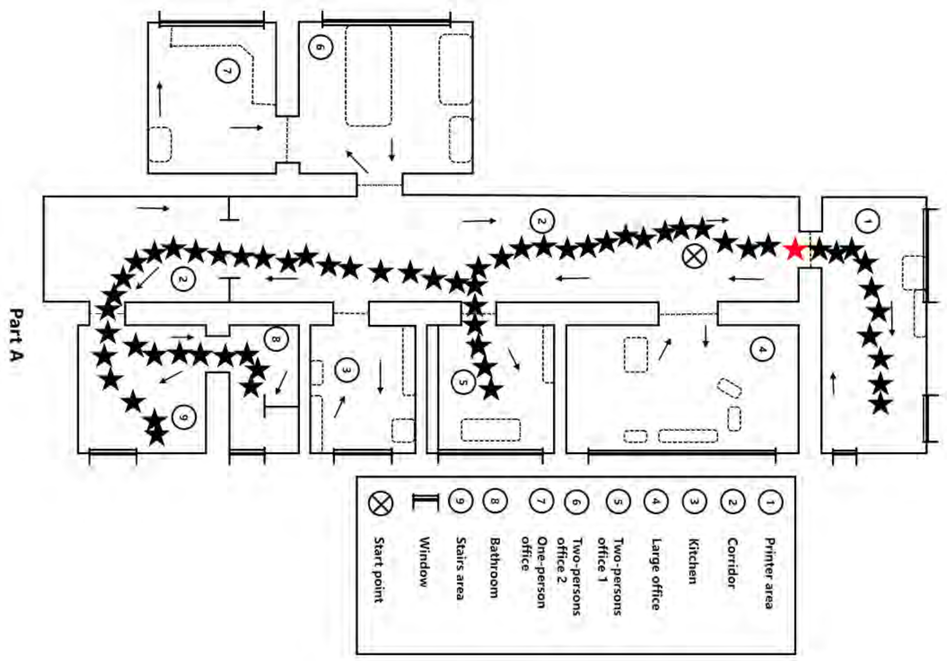


Figure 11.8: A sample recognition task Freiburg standard cloudy. The overall accuracy for the shown sequence is 98.95%.

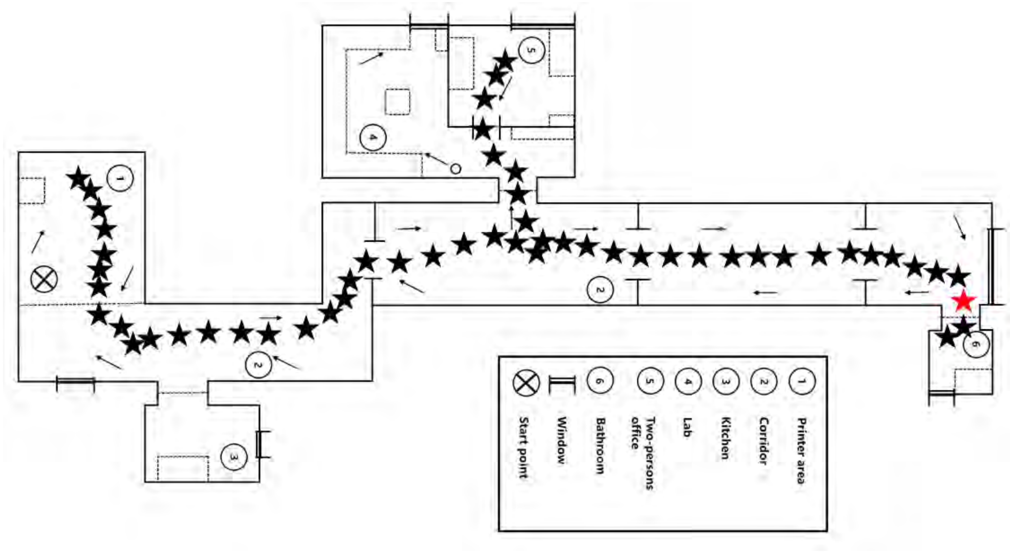


Figure 11.9: A sample recognition task Ljubljana standard night. The overall accuracy for the shown sequence is 98.43%.

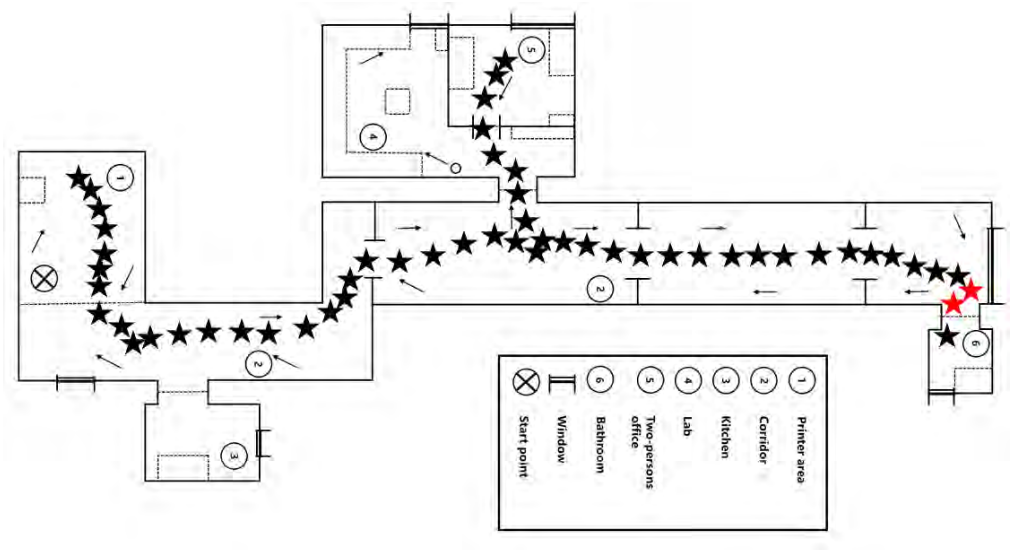


Figure 11.10: A sample recognition task Ljubljana standard sunny. The overall accuracy for the shown sequence is 96.87%.

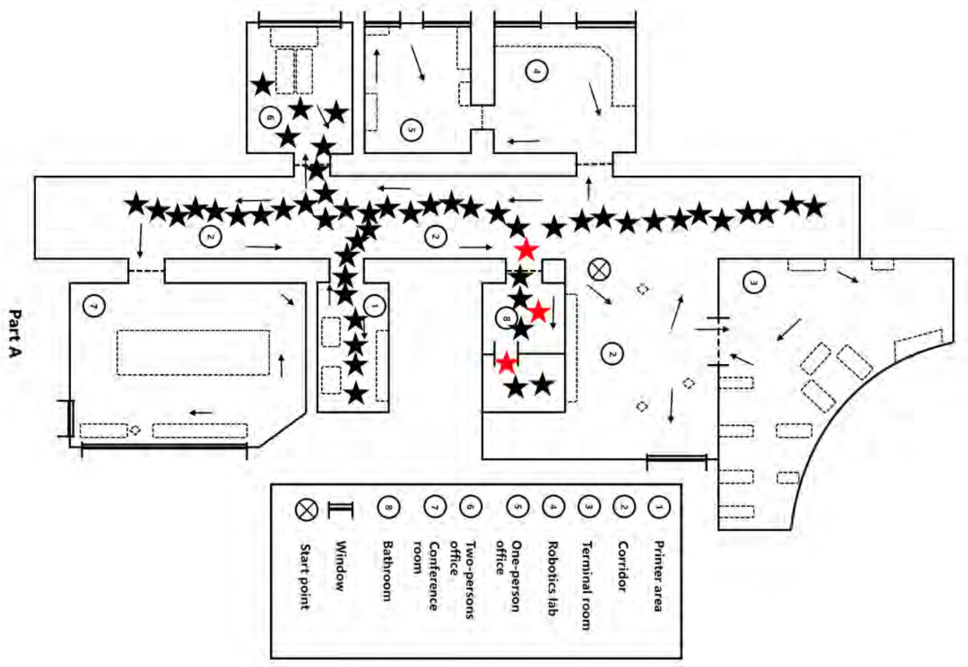


Figure 11.11: A sample recognition task Saarbrücken standard cloudy. The overall accuracy for the shown sequence is 96.87%.

11.6 Conclusion

In this chapter, we proposed a new topological navigation approach based on place recognition method called Convolutional Long Short-Term Memory (C-LSTM). The robot recognized the visited indoor places based only on images which are sequentially captured. Then, it stored and updated automatically the nodes into its incremental topological map. The C-LSTM consisted of Convolution Neural Network (CNN) layers extracting features from the input data and Long Short-Term Memory (LSTM) considering the temporal dependencies of the robot movement. Our place recognition approach and topological mapping provided good results on different indoor environment datasets including those which contain omnidirectional images and also RGB images.

Chapter 12

Conclusion and Perspectives

"The end is never the end. It's always the beginning of something."

Kate Lord Brown

Contents

THE work carried out in this thesis covers important research topics in the field of deep learning which is applied on mobile robotic applications including object classification, scene classification, and topological navigation. This last chapter summarizes the approaches achieved in this thesis. We will first recall the main contributions of this work. We will then discuss the limitations of the proposed approaches, and some future directions and research perspectives. Finally, we will present a list of publications related to the work of this thesis.

12.1 Summary of the contributions

The contributions of this thesis are achieved in the field of mobile robotics using different deep learning architectures. They are mainly subdivided into three parts: object classification, scene classification, and finally topological navigation. For simplicity, this section is also organized into three parts.

Part I Object classification

Object recognition is a fundamental task for a large number of mobile robotic applications such as object grasping, scene recognition, or Simultaneous Localization and Mapping (SLAM). In this part, we have proposed several contributions that are essentially based on the extraction of 3D keypoints and descriptors from 3D point clouds, then learning them using different types of Deep Belief Networks (DBNs). The first contribution (Chapter 3) proposed a new 3D recognition pipeline based on PCL's descriptors as well as the recognition threshold to perform the object recognition task. We acquired a new dataset of 3D real-world objects which are captured from multiple Kinect frames using both RGBDemo software and our mobile hardware. Then, we evaluated the 3D descriptors implemented in the PCL library by proposing a new 3D recognition pipeline which used a recognition threshold rejecting misclassified objects. The experimental results show that our proposed pipeline is able to produce good results comparing with the Alexandre's work [5]. Also, we confirmed that our 3D shaped objects of different view acquisition are relevant than those acquired from a single Kinect frame. The second contribution (Chapter 4) suggested several approaches for 2D/3D object categorization and recognition. Firstly, we described 2D object database and 3D point clouds with 2D Speeded-Up Robust Features (2D SURF) and Spin Images (SI) descriptors respectively, which are quantified using the k-means clustering algorithm in order to obtain the 2D/3D Bag of Words (2D/3D BoWs). Then, we proposed a new global descriptor called "VFH-Color" that combined geometric features extracted from the previous Viewpoint Feature Histogram (VFH) descriptor and color information extracted from the color quantization method. After that, we learned the extracted 2D/3D features with DBN. The experimental results on ALOI and Washington RGBD datasets clearly ascertain that the proposed algorithms are able to classify images and 3D point clouds. These results are encouraging, especially that our new VFH-Color descriptor performed the state-of-the-art methods in recognizing 3D objects under different views. Also, our approach improved the recognition rates thanks to the use of color information. The accuracy using VFH-Color performs 3% better than VFH that models only the geometric features. In the last contribution of this part (chapter 5), we proposed a global approach for representing and learning 3D object categories using global descriptor and deep learning architectures. As global descriptors describe an entire object, a pre-processing step is usually required to remove planes and walls in the 3D scene and then segment it into different objects. For that, we segmented objects from 3D laboratory scenes using Euclidean cluster extraction algorithm. The majority of the objects are well segmented except some ones which didn't have an important height are confused with the planar surface model and are not selected as segmentation candidates. After the segmentation step, we extracted geometric features from 3D point clouds using the

VFH descriptor and then we learned these features with DBN. Thereafter, we evaluated the performance of both Generative and Discriminative DBN architectures (GDBN/DDBN) for 3D object categorization task using different RBM training techniques which include Contrastive Divergence (CD), Persistent Contrastive Divergence (PCD), and Free Energy in Persistent Contrastive Divergence (FEPCD). GDBN trained a sequence of Restricted Boltzmann Machines (RBMs) while DDBN used a new deep architecture based on RBMs and the joint density model. The experimental results using DDBN with FEPCD training method are encouraging, especially that our approach is able to classify 3D objects under different views with accuracy value 96.43%. Also, discriminative training contrary to generative one holds the promise of learning powerful end to end systems given enough labeled training data.

Part II Scene classification The scene classification problem is one of the most difficult challenges in computer vision and robotics. In this part, we proposed two contributions that showed very important results compared to the state-of-the-art 2D and 3D scene classification. In the first contribution (chapter 7), we focused on biologically inspired methods for representation and classification of indoor environments. First, we extracted the global visual features from GIST descriptor that operates like the human visual system by shortly extracting the essential information in the image regardless of its complexity. Then, we learned these resulting features using DDBN which is stimulated by the biological depth of brain and provided discriminative power for pattern classification. DDBN employed a new deep architecture which is based on Restricted Boltzmann Machines (RBMs) and the joint density model. The backpropagation technique is used over the entire classifier to fine-tune the weights for an optimum classification. Moreover, the objective of this contribution was to ensure a classification system that performs as Convolution Neural Network (CNN) in term of recognition rate but requires a short computing time. The performances of the combination of GIST descriptor and DDBN in term of computational complexity remains better than CNN since the CNN used many layers (i.e., convolution, max-pooling, and fully-connected) to extract then classify the features. The experimental results clearly ensured that the proposed algorithm can classify indoor environments with almost the same accuracy as CNN, but outperforms it in terms of computational efficiency. In the second contribution (Chapter 8), we proposed a novel multimodal neural network architecture for RGBD indoor scene classification based on a simple depth encoding approach and pre-trained CNN. Our architecture consisted of two separate CNNs trained on RGB and depth images, then combined with a late fusion network. Thereby, we introduced a simple depth colorization method in order to use depth images as inputs for the CNNs. Then, we learned RGB and colorized depth images separately using pre-trained CNNs on Places dataset, followed by a third training step in which, the architecture fine-tuned two modalities with a fusion network that performed the final classification. Finally, we carried out a set of experiments using different scenarios and achieved the state-of-the-art performance on the NYU V1 dataset with accuracy value 93.19 % and NYU V2 data with the accuracy value 85.33 %.

Part III Topological navigation Navigation, especially mapping, is a critical task for autonomous mobile robots. In this last part, we have developed a new concept of incremental topological mapping. The first contribution (Chapter 10) presented a new method of exploring indoor environments by an autonomous mobile robot, as well as building topological maps based on global visual attributes. This approach included the advantage of using the small size of the GIST descriptors and the ease of their calculation, as well as the advantages of omnidirectional images with a horizontal field of 360° views. Furthermore, in order to handle the problem of a visual loop closing, we employed a statistical formula that correctly assigned each global descriptor to its right location. Our experimental results are very promising thanks to the quality of the environment representation which is based on compact and fast global features. Also, the results showed the robustness of the loop closing detection which used a statistical formula, thus building a coherent topological map. In the second contribution, we proposed a new topological navigation approach based on place recognition method called Convolutional Long Short-Term Memory (C-LSTM). The C-LSTM involved CNN layers to extract features from the input data combined with LSTM to consider the information of the previous frames and to learn the temporal dependencies of the robot movement. For feature extraction, we used the pre-trained CNN on the scene-centric dataset called Places, then the extracted features are fed to the LSTM input. After performing the place recognition, the robot created and updated its topological map in order to act in its environment. The robot recognized the visited indoor places based only on images which are sequentially captured. Then, it stored and updated automatically the nodes into its incremental topological map. Our place recognition approach and topological mapping provided good results on different indoor environment datasets including those which contain omnidirectional images and also RGB images.

12.2 Limitation and perspectives

The results obtained during this thesis are globally very promoting and encouraging. However, in this section, we focus on the limitation of some approaches and the problems encountered throughout these work, and, at the same time, the possible solutions to overcome them. By this way, we propose the directions for future research area which have not been explored in the present dissertation.

Training deep neural networks with large datasets requires an increasing amount of computation resources. This might take from hours to weeks depending on the dataset, the computational power, and the algorithms being used for the training. However, the common limitation of all our approaches depends on the hardware used to learn our data. In our experiments, we used only the CPU device because of the limited graphic memory of our GPU card. Therefore, we fixed a limited number of epochs and small size of image datasets, which may influence the obtained results. In future work, we will implement our approaches on GPU card to be massively parallelized and thus sped up.

In the first part, we also encountered the problem of 3D object segmentation using Euclidean cluster extraction algorithm which consisted of two major steps: (i) plane surface segmentation, and (ii) object segmentation. Nevertheless, some objects that are mingled in the planar surface model (e.g., table) such as a plate, a small flashlight, and a magazine are not segmented since they are considered by the algorithm as part of the planar surface model. In future work, we attempt to develop a new 3D segmentation algorithm based on PCL libraries in order to face this limitation. Since our 2D/3D object classification approaches showed good results compared with the state-of-the-art and are able to classify environment objects, we will extend our work to object grasping which constitutes an essential component in an autonomous robotic manipulation system operating in human environments.

In the second part, we will exploit the object classification results to perform indoor scene recognition through the objects present in the scene. We will assign a probability to each object class, then count all the object probabilities in order to predict the scene class. By this way, the object and scene classification will be two dependent tasks which can be used in the mobile robotic navigation.

In the last part, we will propose a semantic navigation approach based on the sequence to sequence learning. Such an approach will provide a high-level communication between robots and humans. Besides omnidirectional and RGB images, in the next work, we will integrate depth information to perform navigation with RGBD sensors. Moreover, we will attempt to develop a bio-inspired navigation method from insect and equidae navigation which can be used in both indoor and outdoor environments.

Deep learning methods are spreading rapidly in several domains such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, and machine translation. Recently, we incorporated deep learning methods in both dynamic texture for outdoor scene classification [296] and emotion recognition framework [289] which are not reported in this dissertation since these work are slightly different from the approaches presented here. The ultimate aim is to combine all these approaches to embed them in a mobile robot in order to ensure a successful and a global human-robot interaction project.

12.3 List of publications

The work presented in this thesis produced a series of publications published at international journals, book chapters, and conferences in the field.

12.3.1 International journals

- **ZRIRA Nabila**, KHAN Haris Ahmad, and BOUYAKHF El Houssine. *Discriminative Deep Belief Network for Indoor Environment Classification Using Global Visual Features.* Cognitive Computation, 2018, vol. 10, no 3, p. 437-453.

- **ZRIRA Nabila** and BOUYAKHF El Houssine. *A novel incremental topological mapping using global visual features.* International Journal of Computational Vision and Robotics, 2018, vol. 8, no 1, p. 18-31.

- OUADIAY Fatima Zahra, **ZRIRA Nabila**, HANNAT Mohamed, BOUYAKHF El Houssine, and HIMMI Majid. *3D object classification based on deep belief networks and point clouds.* International Journal of Computational Vision and Robotics, 2019, In press.

- **ZRIRA Nabila**, AHMAD Saquib, HANNAT Mohamed, MARTINS MARINHO João Paulo Virgílio, and BOUYAKHF El Houssine. *Multimodal Feature Fusion for Robust RGBD Indoor Scene Classification.* IET Computer Vision, submitted.

- **ZRIRA Nabila**, and BOUYAKHF El Houssine. *End to End Learning to Navigate in Indoor Envrionment.* The International Journal of Robotics Research, submitted.

12.3.2 Book chapters

- **ZRIRA Nabila**, HANNAT Mohamed, BOUYAKHF El Houssine, and KHAN Haris Ahmed. *2D/3D Object Recognition and Categorization Approaches for Robotic Grasping.* In : Advances in Soft Computing and Machine Learning in Image Processing. Springer, Cham, 2018. p. 567-593.

- **ZRIRA Nabila**, HANNAT Mohamed, and BOUYAKHF El Houssine. *3D Object Categorization in Cluttered Scene using Deep Belief Network Architectures.* In: Nature-Inspired Computation in Data Mining and Machine Learning. Springer.

12.3.3 Conferences and Workshops

- **ZRIRA Nabila**, HANNAT Mohamed, and BOUYAKHF El Houssine. *VFH-Color and Deep Belief Network for 3D Point Cloud Recognition.* In : Iberian Conference on Pattern Recognition and Image Analysis. Springer, Cham, 2017. p. 445-452.

- **ZRIRA Nabila**, HANNAT Mohamed, BOUYAKHF El-Houssine, and KHAN Haris Ahmed. *Generative vs. Discriminative Deep Belief Netwok for 3D Object Categorization.* In : VISIGRAPP (5: VISAPP). 2017. p. 98-107.

- **ZRIRA Nabila**, OUADIAY Fatima Zahra, HANNAT Mohamed, BOUYAKHF El Houssine, and HIMMI Majid. *Evaluation of PCL's Descriptors for 3D Object Recognition in Cluttered Scene.* In : Proceedings of the 2nd international Conference on Big Data, Cloud and Applications. ACM, 2017. p. 81.

- OUADIAY Fatima Zahra, **ZRIRA Nabila**, BOUYAKHF El Houssine, and HIMMI Majid. *3d object categorization and recognition based on deep belief networks and point clouds.* In : Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics. SCITEPRESS-Science and Technology Publications, Lda, 2016. p. 311-318.

- HANNAT Mohamed, **ZRIRA Nabila**, RAOUI Younès, and BOUYAKHF El Houssine. *A fast object recognition and categorization technique for robot grasping using the visual bag of words.* In : Multimedia Computing and Systems (ICMCS), 2016 5th International Conference on. IEEE, 2016. p. 173-178.

- **ZRIRA Nabila**, RAOUI Younès, BOUYAKHF El-Houssine, and AITFARES Wassima. *Topological navigation for mobile robot in indoor environment using global visual features.* In World Conference on Complex Systems (WCCS), 2014.

- **ZRIRA Nabila**, RAOUI Younès, BOUYAKHF El-Houssine, and AITFARES Wassima. *Topological navigation of mobile in indoor environment.* In Robotics Workshop: Trends and Challenges, 2014.

Bibliography

[1] Ala Aboudib, Vincent Gripon, and Gilles Coppin. A biologically inspired framework for visual information processing and an application on modeling bottom-up visual attention. *Cognitive Computation*, pages 1–20, 2016. 112, 113

[2] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985. 23

[3] A. Aldoma, F. Tombari, R.B. Rusu, and M. Vincze. *OUR-CVFH–Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation*. Springer, 2012. 55

[4] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R.B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 585–592. IEEE, 2011. 55

[5] L.A. Alexandre. 3d descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*, volume 1. Citeseer, 2012. 47, 59, 191

[6] Luís A Alexandre. 3d object recognition using convolutional neural networks with transfer learning between input channels. In *Intelligent Autonomous Systems 13*, pages 889–898. Springer, 2016. 45

[7] Khaled Alhamzi, Mohammed Elmogy, and Sherif Barakat. 3d object recognition based on local and global features using point cloud library. *International Journal of Advancements in Computing Technology*, 7(3):43, 2015. 47

[8] John Robert Anderson. *Cognitive psychology and its implications*. WH Freeman/Times Books/Henry Holt & Co, 1985. 99

[9] Ion Androutsopoulos, John Koutsias, Konstantinos V Chandrinos, and Constantine D Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM, 2000. 3

[10] Adrien Angeli, Stéphane Doncieux, Jean-Arcady Meyer, and David Filliat. Visual topological slam and global localization. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4300–4305. IEEE, 2009. 151

[11] Sandra Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, and A de A Araújo. Bossa: Extended bow formalism for image classification. In *2011 18th IEEE International Conference on Image Processing*, pages 2909–2912. IEEE, 2011. 65

[12] Jing Bai, Jian-Yun Nie, and François Paradis. Using language models for text classification. In *Proceedings of the Asia Information Retrieval Symposium, Beijing, China*, 2004. 42

[13] Dana H Ballard, Mary M Hayhoe, and Jeff B Pelz. Memory representations in natural tasks. *Journal of Cognitive Neuroscience*, 7(1):66–80, 1995. 102

[14] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 65

[15] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006. 65, 128

[16] Patrick Beeson, Nicholas K Jong, and Benjamin Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *IEEE International Conference on Robotics and Automation*, volume 4, page 4373. IEEE; 1999, 2005. 156

[17] Yoshua Bengio, Nicolas Chapados, Olivier Delalleau, Hugo Larochelle, Xavier Saint-Mleux, Christian Hudon, and Jérôme Louradour. Detonation classification from acoustic signature with the restricted boltzmann machine. *Computational Intelligence*, 28(2):261–288, 2012. xiv, 26

[18] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 4

[19] Yoshua Bengio, Olivier Delalleau, and Nicolas L Roux. The curse of highly variable functions for local kernel machines. In *Advances in neural information processing systems*, pages 107–114, 2006. 6

[20] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 22, 113

[21] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007. 7

[22] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007. 6

[23] Irving Biederman. *On the semantics of a glance at a scene*. 1981. 112

[24] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 63

[25] Josep Miquel Biosca and José Luis Lerma. Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):84–98, 2008. 83

[26] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Depth kernel descriptors for object recognition. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 821–826. IEEE, 2011. 44, 78, 79

[27] Dmytro Bobkov, Sili Chen, Ruiqing Jian, Zafar Iqbal, and Eckehard Steinbach. Noise-resistant deep learning for object classification in 3d point clouds using a point pair descriptor. *IEEE Robotics and Automation Letters*, 2018. 46

[28] Anastasia Bolovinou, Ioannis Pratikakis, and S Perantonis. Bag of spatio-visual words for context inference in scene classification. *Pattern Recognition*, 46(3):1039–1053, 2013. 42, 106

[29] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):185–207, 2013. xv, 100

[30] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Scene classification using a hybrid generative/discriminative approach. *IEEE transactions on pattern analysis and machine intelligence*, 30(4):712–727, 2008. 128

[31] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992. 70

[32] Somar Boubou, Hamed Jabbari Asl, Tatsuo Narikiyo, and Michihiro Kawanishi. Real-time recognition and pursuit in robots based on 3d depth data. *Journal of Intelligent & Robotic Systems*, pages 1–14, 2018. 44

[33] David J Burr. Experiments on neural net recognition of spoken and written text. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1162–1168, 1988. 12

[34] Ziyun Cai and Ling Shao. Rgb-d scene classification via multi-modal feature learning. *Cognitive Computation*, pages 1–16, 2018. 109

[35] Gail A Carpenter and William D Ross. Art-emap: A neural network architecture for object recognition by evidence accumulation. *Neural Networks, IEEE Transactions on*, 6(4):805–818, 1995. 113

[36] Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer, 2005. 29

[37] Ioannis Cassagne, Nicolas Riche, Marc Décombas, Matei Mancas, Bernard Gosselin, Thierry Dutoit, and Robert Laganiere. Video saliency based on rarity prediction: Hyperaptor. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 1521–1525. IEEE, 2015. 101

[38] Alexandre Chapoulie, Patrick Rives, and David Filliat. Topological segmentation of indoors/outdoors sequences of spherical views. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4288–4295. IEEE, 2012. 154

[39] Antonio Chella, Irene Macaluso, and Lorenzo Riano. Automatic place detection and localization in autonomous robotics. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 741–746. IEEE, 2007. 156

[40] Hongtai Cheng, Heping Chen, and Yong Liu. Topological indoor localization and navigation for autonomous mobile robot. *IEEE Transactions on Automation Science and Engineering*, 12(2):729–738, 2015. 152

[41] Laura A Clemente, Andrew J Davison, Ian D Reid, José Neira, and Juan D Tardós. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*, volume 2, 2007. 165

[42] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004. 43

[43] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008. 151, 165

[44] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 124

[45] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3, 2014. 6, 22

[46] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014. 22, 113

[47] Robert Desimone, Stanley J Schein, Jeffrey Moran, and Leslie G Ungerleider. Contour, color and shape analysis beyond the striate cortex. *Vision research*, 25(3):441–452, 1985. 5

[48] Mandar Dixit, Si Chen, Dashan Gao, Nikhil Rasiwasia, and Nuno Vasconcelos. Scene classification with semantic fisher vectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2974–2983, 2015. 107

[49] Tom Downs, Kevin E Gates, and Annette Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2(Dec):293–297, 2001. 154

[50] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE, 2015. 45, 78, 79, 95, 96, 124, 130

[51] Özgür Erkent and H Işıl Bozma. Bubble space and place representation in topological maps. *The International Journal of Robotics Research*, 32(6):672–689, 2013. 185, 186

[52] Özgür Erkent and H Işıl Bozma. Long-term topological place learning. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5462–5467. IEEE, 2015. 156

[53] Ehsan Fazl-Ersi, James H Elder, and John K Tsotsos. Hierarchical classifiers for robust topological robot localization. *Journal of Intelligent & Robotic Systems*, 68(2):147–163, 2012. 153

[54] Ziyong Feng, Lianwen Jin, Dapeng Tao, and Shuangping Huang. Dlanet: A manifold-learning-based discriminative feature learning network for scene classification. *Neurocomputing*, 157:11–21, 2015. 108, 141, 142, 182, 183

[55] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–264. IEEE, 2003. 43

[56] David Filliat. A visual bag of words method for interactive qualitative localization and mapping. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3921–3926. IEEE, 2007. 42

[57] Asja Fischer and Christian Igel. Training restricted boltzmann machines: An introduction. *Pattern Recognition*, 47(1):25–39, 2014. 23

[58] Marco Fornoni and Barbara Caputo. Indoor scene recognition using task and saliency-driven feature pooling. In *Proceedings of the British Machine Vision Conference*, number EPFL-CONF-192418, 2012. 103

[59] Wolfgang Förstner, Timo Dickscheid, and Falko Schindler. Detecting interpretable and accurate scale-invariant keypoints. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2256–2263. IEEE, 2009. 128

[60] Alinda Friedman. Framing pictures: The role of knowledge in automatized encoding and memory for gist. *Journal of experimental psychology: General*, 108(3):316, 1979. 115

[61] Charles R Gallistel. *The organization of learning*. The MIT Press, 1990. 145

[62] José Gaspar, Niall Winters, and José Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on robotics and automation*, 16(6):890–898, 2000. 153

[63] Jan-Mark Geusebroek, Gertjan J Burghouts, and Arnold WM Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005. 71

[64] Toon Goedemé, Marnix Nuttin, Tinne Tuytelaars, and Luc Van Gool. Omnidirectional vision based topological navigation. *International Journal of Computer Vision*, 74(3):219–236, 2007. 151

[65] Aleksey Golovinskiy and Thomas Funkhouser. Min-cut based segmentation of point clouds. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 39–46. IEEE, 2009. 83

[66] F Gomez-Donoso, A Garcia-Garcia, J Garcia-Rodriguez, S Orts-Escolano, and M Cazorla. Lonchanet: A sliced-based cnn architecture for real-time 3d object recognition. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 412–418. IEEE, 2017. 46

[67] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 3

[68] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013. 108, 141, 142, 182, 183

[69] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 3, 2017. 157

[70] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014. 124, 130

[71] Amin Haji-Abolhassani and James J Clark. An inverse yarbus process: Predicting observers' task from eye movement patterns. *Vision research*, 103:127–142, 2014. xv, 102

[72] Mohamed Hannat, Nabila Zrira, Younès Raoui, and El Houssine Bouyakhf. A fast object recognition and categorization technique for robot grasping using the visual bag of words. In *Multimedia Computing and Systems (ICMCS), 2016 5th International Conference on*, pages 173–178. IEEE, 2016. 43, 63

[73] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988. 128

[74] Munawar Hayat, Salman H Khan, Mohammed Bennamoun, and Senjian An. A spatial layout and scale invariant feature representation for indoor scene classification. *arXiv preprint arXiv:1506.05532*, 2015. 107, 141, 142, 182, 183

[75] Mary Hayhoe and Dana Ballard. Eye movements in natural behavior. *Trends in cognitive sciences*, 9(4):188–194, 2005. 99, 102

[76] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 133

[77] Vishakh Hegde and Reza Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016. 46

[78] Michael H Herzog and Aaron M Clarke. Why vision is not both hierarchical and feedforward. *Frontiers in computational neuroscience*, 8, 2014. xiv, 5, 6

[79] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. 28, 29, 86

[80] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012. 26

[81] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 22, 29, 30, 113

[82] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 22, 25

[83] Geoffrey E Hinton and Ruslan R Salakhutdinov. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pages 1607–1614, 2009. 27

[84] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 36

[85] Fan Hu, Gui-Song Xia, Zifeng Wang, Xin Huang, Liangpei Zhang, and Hong Sun. Unsupervised feature learning via spectral clustering of multidimensional patches for remotely sensed scene classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(5), 2015. 65, 106

[86] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017. 133

[87] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998. 101, 112

[88] W James. The principles of psychology, vol. 2. ny, us: Henry holt and company, 1890. 99

[89] Lee Jiann-Der. Object recognition using a neural network with optimal feature extraction. *Mathematical and Computer Modelling*, 25(12):105–117, 1997. 113

[90] Vasu Jindal. Generating image captions in arabic using root-word based recurrent neural networks and deep neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 144–151, 2018. 35

[91] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999. 54, 67

[92] Olivier R Joubert, Guillaume A Rousselet, Denis Fize, and Michèle Fabre-Thorpe. Processing scene context: Fast categorization and object interference. *Vision research*, 47(26):3286–3297, 2007. 103

[93] Mayank Juneja, Andrea Vedaldi, CV Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 923–930, 2013. 106

[94] Barry L Kalman and Stan C Kwasny. Why tanh: choosing a sigmoidal function. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 4, pages 578–581. IEEE, 1992. 18

[95] Hakan Karaoguz and H Işıl Bozma. Reliable topological place detection in bubble space. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 697–702. IEEE, 2014. 156

[96] Hakan Karaoğuz and H Işıl Bozma. An integrated model of autonomous topological spatial cognition. *Autonomous Robots*, 40(8):1379–1402, 2016. 156

[97] S Hamidreza Kasaei, Juil Sock, Luıs Seabra Lopes, Ana Maria Tomé, and Tae-Kyun Kim. Perceiving, learning, and recognizing 3d objects: An approach to cognitive service robots. 2018. 44

[98] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004. 42

[99] Sami Keronen, KyungHyun Cho, Tapani Raiko, Alexander Ilin, and Kalle Palomäki. Gaussian-bernoulli restricted boltzmann machines and automatic feature extraction for noise robust missing data mask estimation. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6729–6733. IEEE, 2013. 25

[100] Mohammad Ali Keyvanrad and Mohammad Mehdi Homayounpour. Deep belief network training improvement using elite samples minimizing free energy. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(05):1551006, 2015. 29, 31, 86

[101] Rahat Khan, Cécile Barat, Damien Muselet, and Christophe Ducottet. Spatial orientations of visual word pairs to improve bag-of-visual-words model. In *Proceedings of the British Machine Vision Conference*, pages 89–1. BMVA Press, 2012. 43

[102] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Roberto Togneri, and Ferdous A Sohel. A discriminative representation of convolutional features for indoor scene recognition. *IEEE Transactions on Image Processing*, 25(7):3372–3383, 2016. 107, 141, 142, 182, 183

[103] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on neural networks*, 3(6):962–968, 1992. 12

[104] Christof Koch and Shimon Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. In *Matters of intelligence*, pages 115–141. Springer, 1987. 100

[105] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 132

[106] Norbert Kruger, Peter Janssen, Sinan Kalkan, Markus Lappe, Ales Leonardis, Justus Piater, Antonio J Rodriguez-Sanchez, and Laurenz Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1847–1871, 2013. 5

[107] Benjamin Kuipers, Rob Browning, Bill Gribble, Mike Hewett, and Emilio Remolina. The spatial semantic hierarchy. *Artificial intelligence*, 2000. 162

[108] Ankita Kumar, Jean-Philippe Tardif, Roy Anati, and Kostas Daniilidis. Experiments on visual loop closing using vocabulary trees. 2008. 153

[109] Matthias Kümmerer, Thomas SA Wallis, and Matthias Bethge. Deepgaze ii: Reading fixations from deep features trained on object recognition. *arXiv preprint arXiv:1610.01563*, 2016. 101

[110] Mathieu Labbe and Francois Michaud. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745, 2013. 155

[111] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011. xiv, 44, 49, 71, 78, 79, 86, 95, 96

[112] Michael F Land and Sophie Furneaux. The knowledge base of the oculomotor system. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 352(1358):1231–1239, 1997. 102

[113] Michael F Land and David N Lee. Where we look when we steer. *Nature*, 369(6483):742, 1994. 102

[114] Michael F Land, N Mennie, and J Rusted. Eye movements and the roles of vision in activities of daily living: making a cup of tea. *Perception*, 28(4):1311–1328, 1999. 102

[115] Diane Larlus, Jakob Verbeek, and Frédéric Jurie. Category level object segmentation by combining bag-of-words models with dirichlet processes and random fields. *International Journal of Computer Vision*, 88(2):238–253, 2010. 43

[116] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM, 2008. 26

[117] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997. 12

[118] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006. 105, 119

[119] Yann Le Cun, LD Jackel, B Boser, JS Denker, HP Graf, Isabelle Guyon, Don Henderson, RE Howard, and W Hubbard. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989. 32

[120] Olivier Le Meur, Patrick Le Callet, Dominique Barba, and Dominique Thoreau. A coherent computational approach to model the bottom-up visual attention. *IEEE transactions on pattern analysis and machine intelligence*, 28:802–817, 2006. 112, 113

[121] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990. 12

[122] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 132

[123] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, UA Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995. 12

[124] Daniel T Levin and Daniel J Simons. Failure to detect changes to attended objects in motion pictures. *Psychonomic Bulletin & Review*, 4(4):501–506, 1997. 99

[125] Esther Levin. Word recognition using hidden control neural architecture. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 433–436. IEEE, 1990. 12

[126] Fei Fei Li, Rufin VanRullen, Christof Koch, and Pietro Perona. Rapid natural scene categorization in the near absence of attention. *Proceedings of the National Academy of Sciences*, 99(14):9596–9601, 2002. 103

[127] Li-Jia Li, Richard Socher, and Li Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2036–2043. IEEE, 2009. 128

[128] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in neural information processing systems*, pages 1378–1386, 2010. 105

[129] Li-Jia Li, Hao Su, Yongwhan Lim, and Li Fei-Fei. Objects as attributes for scene classification. In *European Conference on Computer Vision*, pages 57–69. Springer, 2010. 105

[130] Mingjing Li, Wei-Ying Ma, Zhiwei Li, and Lei Wu. Visual language modeling for image classification, February 28 2012. US Patent 8,126,274. 42

[131] Teng Li, Tao Mei, In-So Kweon, and Xian-Sheng Hua. Contextual bag-of-words for visual categorization. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(4):381–392, 2011. 42

[132] Yiyi Liao, Sarath Kodagoda, Yue Wang, Lei Shi, and Yong Liu. Understand scene categories by objects: A semantic regularized scene classifier using convolutional neural networks. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2318–2325. IEEE, 2016. 109

[133] Huei-Yung Lin, Yu-Hsiang Lin, and Jia-Wei Yao. Scene change detection and topological map construction using omnidirectional image sequences. In *MVA*, pages 57–60, 2013. 155, 185, 186

[134] Huei-Yung Lin, Chia-Wei Yao, Kai-Sheng Cheng, et al. Topological map construction and scene recognition for vehicle localization. *Autonomous Robots*, 42(1):65–81, 2018. 155

[135] Shang-Hung Lin, Sun-Yuan Kung, and Long-Ji Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE transactions on neural networks*, 8(1):114–132, 1997. 12

[136] Dimitri A Lisin, Marwan A Mattar, Matthew B Blaschko, Erik G Learned-Miller, and Mark C Benfield. Combining local and global image features for object class recognition. In *Computer vision and pattern recognition-workshops, 2005. CVPR workshops. IEEE Computer society conference on*, pages 47–47. IEEE, 2005. 42

[137] Ming Liu and Roland Siegwart. Topological mapping and scene recognition with lightweight color descriptors for an omnidirectional camera. *IEEE Transactions on Robotics*, 30(2):310–324, 2014. 155

[138] Yan Liu, Shusen Zhou, and Qingcai Chen. Discriminative deep belief networks for visual data classification. *Pattern Recognition*, 44(10):2287–2296, 2011. 31

[139] Yang Liu and Hong Zhang. Indexing visual features: Real-time loop closure detection using a tree structure. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3613–3618. IEEE, 2012. 152

[140] Mohammad Reza Loghmani, Mirco Planamente, Barbara Caputo, and Markus Vincze. Recurrent convolutional fusion for rgb-d object recognition. *arXiv preprint arXiv:1806.01673*, 2018. 46

[141] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 65, 67, 128

[142] Xiaoqiang Lu, Xuelong Li, and Lichao Mou. Semi-supervised multitask learning for scene recognition. *IEEE transactions on cybernetics*, 45(9):1967–1976, 2015. 109

[143] David MacKay. Failures of the one-step learning algorithm. In *Available electronically at http://www. inference. phy. cam. ac. uk/mackay/abstracts/gbm. html*, 2001. 29

[144] Lorand Madai-Tahy, Sebastian Otte, Richard Hanten, and Andreas Zell. Revisiting deep convolutional neural networks for rgb-d based object recognition. In *International Conference on Artificial Neural Networks*, pages 29–37. Springer, 2016. 45, 78, 79, 95, 96, 130

[145] Marianna Madry, Carl Henrik Ek, Renaud Detry, Kaiyu Hang, and Danica Kragic. Improving generalization for 3d object categorization with global structure histograms. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1379–1386. IEEE, 2012. 44

[146] Mauro Manassi, Bilge Sayim, and Michael H Herzog. When crowding of crowding leads to uncrowdingshort title?? *Journal of Vision*, 13(13):10–10, 2013. 5

[147] Sophie Marat, Tien Ho Phuoc, Lionel Granjon, Nathalie Guyader, Denis Pellerin, and Anne Guérin-Dugué. Modelling spatio-temporal saliency to predict gaze direction for short videos. *International journal of computer vision*, 82(3):231–243, 2009. 112, 113

[148] Romain Marie, Ouiddad Labbani-Igbida, and El Mustapha Mouaddib. Autonomous exploration and topographical cartography in unknown environment referring to omnidirectional vision. *Traitement du Signal*, 31(1-2):221–243, 2014. 159, 165

[149] Stephen Marsland. *Machine learning: an algorithmic perspective.* Chapman and Hall/CRC, 2011. 20

[150] Stephen Marsland. *Machine learning: an algorithmic perspective.* CRC press, 2015. 3

[151] Jesus Martínez-Gómez, Vicente Morell, Miguel Cazorla, and Ismael García-Varea. Semantic localization in the pcl library. *Robotics and Autonomous Systems*, 75:641–648, 2016. 47

[152] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. 128

[153] Oliver Mattausch, Daniele Panozzo, Claudio Mura, Olga Sorkine-Hornung, and Renato Pajarola. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*, volume 33, pages 11–21. Wiley Online Library, 2014. 129

[154] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 45

[155] Kieran Richard Mc Donald. *Discrete language models for video retrieval.* PhD thesis, Dublin City University, 2005. 42

[156] Timothy P McNamara. Mental representations of spatial relations. *Cognitive psychology*, 18(1):87–121, 1986. 174

[157] Luigi Federico Menabrea and Ada King Countess of Lovelace. *Sketch of the Analytical Engine Invented by Charles Babbage, Esq.* Richard and John E. Taylor, 1843. 12

[158] Enrique Romero Merino, Ferran Mazzanti Castrillejo, Jordi Delgado Pin, and David Buchaca Prats. Weighted contrastive divergence. *arXiv preprint arXiv:1801.02567*, 2018. 29

[159] Jean-Arcady Meyer and David Filliat. Map-based navigation in mobile robots:: Ii. a review of map-learning and path-planning strategies. *Cognitive Systems Research*, 4(4):283–317, 2003. 175

[160] Ajmal Mian, Mohammed Bennamoun, and Robyn Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010. 44

[161] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach.* Springer Science & Business Media, 2013. 3

[162] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Computer Vision—ECCV 2002*, pages 128–142. Springer, 2002. 128

[163] Ruggero Milanese. Detecting salient regions in an image: from biological evidence to computer implementation. *Ph. D. Theses, the University of Geneva*, 1993. 100

[164] Florica Mindru, Tinne Tuytelaars, Luc Van Gool, and Theo Moons. Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding*, 94(1):3–27, 2004. 128

[165] Marvin Minsky and Seymour Papert. Perceptrons. 1969. 15

[166] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015. 133

[167] Ajay K Mishra and Yiannis Aloimonos. Active segmentation. *International Journal of Humanoid Robotics*, 6(03):361–386, 2009. 103

[168] Tom M Mitchell et al. Machine learning. wcb, 1997. 3, 15

[169] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks for phone recognition. In *Nips workshop on deep learning for speech recognition and related applications*, volume 1, page 39, 2009. 22, 113

[170] Abdel-rahman Mohamed and Geoffrey Hinton. Phone recognition using restricted boltzmann machines. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4354–4357. IEEE, 2010. 27

[171] Florent Monay and Daniel Gatica-Perez. On image auto-annotation with latent space models. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 275–278. ACM, 2003. 128

[172] Vinod Nair and Geoffrey E Hinton. 3d object recognition with deep belief nets. In *Advances in Neural Information Processing Systems*, pages 1339–1347, 2009. 45

[173] Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993. 27

[174] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *Robotics, Automation and Mechatronics (RAM), 2013 6th IEEE Conference on*, pages 225–230. IEEE, 2013. 82

[175] Keiller Nogueira, Otávio AB Penatti, and Jefersson A dos Santos. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539–556, 2017. 107

[176] A Oliva. Scene perception. chapter in the new visual neurosciences, eds john s. werner and leo. m. chalupa, 2013. 114

[177] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001. 115, 128

[178] Aude Oliva and Antonio Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006. 115

[179] Fatima Zahra Ouadiay, Nabila Zrira, El Houssine Bouyakhf, and M. Majid Himmi. 3d object categorization and recognition based on deep belief networks and point clouds. In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, pages 311–318, 2016. 45, 63, 113

[180] Megha Pandey and Svetlana Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1307–1314. IEEE, 2011. xviii, 106, 121, 122

[181] James Philbin, Ondřej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 43

[182] Mary C Potter. Meaning in visual search. *Science*, 187(4180):965–966, 1975. 102

[183] Mary C Potter. Short-term conceptual memory for pictures. *Journal of experimental psychology: human learning and memory*, 2(5):509, 1976. 63

[184] Andrzej Pronobis and Barbara Caputo. Cold: The cosy localization database. *The International Journal of Robotics Research*, 28(5):588–594, 2009. 167, 178

[185] Andrzej Pronobis, Luo Jie, and Barbara Caputo. The more you learn, the less you store: Memory-controlled incremental svm for visual place recognition. *Image and Vision Computing*, 28(7):1080–1097, 2010. 154

[186] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 46

[187] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 413–420. IEEE, 2009. 105, 106, 119

[188] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *International archives of photogrammetry, remote sensing and spatial information sciences*, 36(5):248–253, 2006. 82

[189] Jane E Raymond, Kimron L Shapiro, and Karen M Arnell. Temporary suppression of visual processing in an rsvp task: An attentional blink? *Journal of experimental psychology: Human perception and performance*, 18(3):849, 1992. 99

[190] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4791–4796. IEEE, 2012. 129

[191] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007. 113

[192] AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering, 1987. 36

[193] Edmund T Rolls and Gustavo Deco. Attention in natural scenes: neurophysiological and computational bases. *Neural networks*, 19(9):1383–1394, 2006. 99

[194] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 15

[195] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998. 12

[196] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 18, 19

[197] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. MIT press Cambridge, MA, USA:, 1987. 19

[198] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1–6. IEEE, 2009. 83

[199] R.B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 53, 68

[200] R.B. Rusu, N. Blodow, Z.C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. 53

[201] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010. 54, 63, 68

[202] R.B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. 47

[203] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105, 1998. 3

[204] Ruslan Salakhutdinov. Learning deep generative models. *Annual Review of Statistics and Its Application*, 2:361–385, 2015. 25

[205] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007. 28

[206] Angel Domingo Sappa and Michel Devy. Fast range image segmentation by an edge detection strategy. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 292–299. IEEE, 2001. 82

[207] Silvio Savarese and Li Fei-Fei. 3d generic object categorization, localization and pose estimation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 43

[208] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018. 157

[209] Jürgen Schmidhuber. A fixed size storage o (n 3) time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation*, 4(2):243–248, 1992. 36

[210] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007. 83

[211] Max Schwarz, Hannes Schulz, and Sven Behnke. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1329–1335. IEEE, 2015. 45, 78, 79, 95, 96

[212] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia*, pages 357–360. ACM, 2007. 56, 67

[213] Navid Serrano, Andreas Savakis, and A Luo. A computationally efficient approach to indoor/outdoor scene classification. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 146–149. IEEE, 2002. 104

[214] Thomas Serre, Gabriel Kreiman, Minjoon Kouh, Charles Cadieu, Ulf Knoblich, and Tomaso Poggio. A quantitative theory of immediate visual recognition. *Progress in brain research*, 165:33–56, 2007. 22, 113

[215] Jiwon Shin, Rudolph Triebel, and Roland Siegwart. Unsupervised 3d object discovery and categorization for mobile robots. In *Robotics Research*, pages 61–76. Springer, 2017. 44

[216] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013. 108

[217] Christian Siagian and Laurent Itti. Rapid biologically-inspired scene classification using features shared with visual attention. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):300–312, 2007. 103, 104

[218] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 601–608. IEEE, 2011. 108, 136, 177

[219] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 136, 178

[220] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI*, volume 95, pages 1080–1087, 1995. 152

[221] Daniel J Simons and Christopher F Chabris. Gorillas in our midst: Sustained inattentional blindness for dynamic events. *perception*, 28(9):1059–1074, 1999. 99

[222] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 132, 134, 176

[223] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering object categories in image collections. 2005. 42

[224] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003. 42

[225] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673, 2012. 45

[226] Dongjin Song and Dacheng Tao. Biologically inspired feature manifold for scene classification. *IEEE Transactions on Image Processing*, 19(1):174–184, 2010. 103

[227] Shuran Song and Jianxiong Xiao. Tracking revisited using rgbd camera: Unified benchmark and baselines. In *Proceedings of the IEEE international conference on computer vision*, pages 233–240, 2013. 129

[228] Xinhang Song, Luis Herranz, and Shuqiang Jiang. Depth cnns for rgb-d scene recognition: Learning from scratch better than transferring from rgb-cnns. In *AAAI*, pages 4271–4277, 2017. 109, 141, 142, 182, 183

[229] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012. 129

[230] Shiying Sun, Ning An, Xiaoguang Zhao, and Min Tan. A pca–cca network for rgb-d object recognition. *International Journal of Advanced Robotic Systems*, 15(1):1729881417752820, 2018. 47, 95, 96

[231] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991. 128

[232] Nadeem Ahmed Syed, Syed Huan, Liu Kah, and Kay Sung. Incremental learning with support vector machines. 1999. 154

[233] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 132

[234] Martin Szummer and Rosalind W Picard. Indoor-outdoor image classification. In *Content-Based Access of Image and Video Database, 1998. Proceedings., 1998 IEEE International Workshop on*, pages 42–51. IEEE, 1998. 104

[235] L. Tamas and B. Jensen. Robustness analysis of 3d feature descriptors for object recognition using a time-of-flight camera. In *Control and Automation (MED), 2014 22nd Mediterranean Conference of*, pages 1020–1025. IEEE, 2014. 47

[236] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *Asian conference on computer vision*, pages 525–538. Springer, 2012. 44

[237] Dapeng Tao, Lianwen Jin, Zhao Yang, and Xuelong Li. Rank preserving sparse learning for kinect based scene classification. *IEEE transactions on cybernetics*, 43(5):1406–1417, 2013. 108, 129, 141, 142, 182, 183

[238] Fayez Tarsha-Kurdi, Tania Landes, and Pierre Grussenmeyer. Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, volume 36, pages 407–412, 2007. 83

[239] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998. xvi, 148, 159

[240] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 148

[241] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1, 2002. 147

[242] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008. 29, 86

[243] Roberto Toldo, Umberto Castellani, and Andrea Fusiello. A bag of words approach for 3d object categorization. In *Computer Vision/Computer Graphics CollaborationTechniques*, pages 116–127. Springer, 2009. 43

[244] Nicola Tomatis, Illah Nourbakhsh, and Roland Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous systems*, 44(1):3–14, 2003. 152

[245] F. Tombari, S. Salti, and L. D. Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer, 2010. 54

[246] F. Tombari, S. Salti, and L.D. Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 809–812. IEEE, 2011. 54

[247] Antonio Torralba. Modeling global scene factors in attention. *JOSA A*, 20(7):1407–1418, 2003. 102

[248] D Tóvári and N Pfeifer. Segmentation based robust interpolation-a new approach to laser data filtering. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/19):79–84, 2005. 82

[249] Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980. 100

[250] Stefan Treue. Neural correlates of attention in primate visual cortex. *Trends in neurosciences*, 24(5):295–300, 2001. 99

[251] Olivier Trullier, Sidney I Wiener, Alain Berthoz, and Jean-Arcady Meyer. Biologically based artificial navigation systems: Review and prospects. *Progress in neurobiology*, 51(5):483–544, 1997. 145

[252] Chi-Yi Tsai and Shu-Hsiang Tsai. Simultaneous 3d object recognition and pose estimation based on rgb-d images. *IEEE Access*, 2018. 45

[253] Jan Tünnermann and Bärbel Mertsching. Region-based artificial visual attention in space and time. *Cognitive Computation*, 6(1):125–143, 2014. 103

[254] Aleš Ude and Rüdiger Dillmann. Vision-based robot path planning. In *Advances in Robot Kinematics and Computational Geometry*, pages 505–512. Springer, 1994. 128

[255] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1023–1029. Ieee, 2000. 104

[256] Christoffer Valgren, Tom Duckett, Achim Lilienthal, et al. Incremental spectral clustering and its application to topological mapping. 2007. 153

[257] David C Van Essen, Charles H Anderson, Daniel J Felleman, et al. Information processing in the primate visual system: an integrated systems perspective. *Science*, 255(5043):419–423, 1992. 5

[258] David A Rojas Vigo, Fahad Shahbaz Khan, Joost Van de Weijer, and Theo Gevers. The impact of color on bag-of-words based object recognition. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1549–1553. IEEE, 2010. 43

[259] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989. 12

[260] Christian Wallraven, Barbara Caputo, and Arnulf Graf. Recognition with local features: the kernel recipe. In *null*, page 257. IEEE, 2003. 42

[261] Anran Wang, Jianfei Cai, Jiwen Lu, and Tat-Jen Cham. Modality and component aware feature fusion for rgb-d scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5995–6004, 2016. 109, 141, 142, 182, 183

[262] Min-Liang Wang and Huei-Yung Lin. A hull census transform for scene change detection and recognition towards topological map building. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 548–553. IEEE, 2010. 154, 155

[263] Tsun-Hsuan Wang, Hung-Jui Huang, Juan-Ting Lin, Chan-Wei Hu, Kuo-Hao Zeng, and Min Sun. Omnidirectional cnn for visual place recognition and navigation. *arXiv preprint arXiv:1803.04228*, 2018. 156

[264] Yuxia Wang, Qingjie Zhao, Bo Wang, Shixian Wang, Yu Zhang, Wei Guo, and Zhiquan Feng. A real-time active pedestrian tracking system inspired by the human visual system. *Cognitive Computation*, pages 1–13, 2015. 103

[265] Steven D Whitehead and Dana H Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83, 1991. 151

[266] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. An image-to-map loop closing method for monocular slam. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2053–2059. IEEE, 2008. 165

[267] Ronald J Williams and David Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Backpropagation: Theory, architectures, and applications*, 1:433–486, 1995. 36

[268] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2987–2992. IEEE, 2011. 55

[269] Jeremy M Wolfe and W Gray. Guided search 4.0. *Integrated models of cognitive systems*, pages 99–119, 2007. 102

[270] Jianxin Wu and Jim M Rehg. Centrist: A visual descriptor for scene categorization. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1489–1501, 2011. 128

[271] Lei Wu, Steven CH Hoi, and Nenghai Yu. Semantics-preserving bag-of-words models and applications. *Image Processing, IEEE Transactions on*, 19(7):1908–1920, 2010. 43

[272] Xuehan Xiong, Daniel Munoz, J Andrew Bagnell, and Martial Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2609–2616. IEEE, 2011. 83

[273] Takayoshi Yamashita, Masayuki Tanaka, Eiji Yoshida, Yuji Yamauchi, and Hironobu Fujiyoshii. To be bernoulli or to be gaussian, for a restricted boltzmann machine. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 1520–1525. IEEE, 2014. 25

[274] AL Yarbus. Eye movements and vision. 1967. *New York*, 1967. 101

[275] Alan L Yuille. The convergence of contrastive divergences. In *Advances in neural information processing systems*, pages 1593–1600, 2005. 29

[276] Hasan FM Zaki, Faisal Shafait, and Ajmal Mian. Convolutional hypercube pyramid for accurate rgb-d object category and instance recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1685–1692. IEEE, 2016. 95, 96

[277] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 132

[278] Jianming Zhang and Stan Sclaroff. Saliency detection: A boolean map approach. In *Proceedings of the IEEE international conference on computer vision*, pages 153–160, 2013. 101

[279] Meishan Zhang, Yue Zhang, and Duy-Tin Vo. Gated neural networks for targeted sentiment analysis. In *AAAI*, pages 3087–3093, 2016. 35

[280] Wei Zhang, Bing Yu, Gregory J Zelinsky, and Dimitris Samaras. Object class recognition using multiple layer boosting with heterogeneous features. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 323–330. IEEE, 2005. 42

[281] Liang Zheng, Shengjin Wang, Ziqiong Liu, and Qi Tian. Packing and padding: Coupled multi-index for accurate image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1939–1946, 2014. 65

[282] Shuaifeng Zhi, Yongxiang Liu, Xiang Li, and Yulan Guo. Lightnet: a lightweight 3d convolutional neural network for real-time 3d object recognition. In *Eurographics Workshop on 3D Object Retrieval*, 2017. 46

[283] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 689–696. IEEE, 2009. 44

[284] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016. 124

[285] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014. 124

[286] Shusen Zhou, Qingcai Chen, and Xiaolong Wang. Discriminative deep belief networks for image classification. In *2010 IEEE International Conference on Image Processing*, pages 1561–1564. IEEE, 2010. 31

[287] Zhi-Hua Zhou and Min-Ling Zhang. Multi-instance multi-label learning with application to scene classification. In *Advances in neural information processing systems*, pages 1609–1616, 2007. 105

[288] Lei Zhu, Al Bing Rao, and Aldong Zhang. Theory of keyblock-based image retrieval. *ACM Transactions on Information Systems (TOIS)*, 20(2):224–257, 2002. 42

[289] Nabila Zrira, Mehdi Abouzahir, El Houssine Bouyakhf, Ibtissam Benmiloud, and Majid Himmi. Learning combined features for automatic facial expression recognition. *IJKESDP*, 2019. 194

[290] Nabila Zrira and El Houssine Bouyakhf. A novel incremental topological mapping using global visual features. *International Journal of Computational Vision and Robotics*, 8(1):18–31, 2018. 112, 159

[291] Nabila Zrira, Mohamed Hannat, and El Houssine Bouyakhf. Vfh-color and deep belief network for 3d point cloud recognition. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 445–452. Springer, 2017. 63

[292] Nabila Zrira, Mohamed Hannat, and El Houssine Bouyakhf. 3d object categorization in cluttered scene using deep belief network architectures. In *Nature-Inspired Computation in Data Mining and Machine Learning*. Springer, 2019. 81

[293] Nabila Zrira, Mohamed Hannat, El Houssine Bouyakhf, and Haris Ahmad Khan. Generative vs. discriminative deep belief netwok for 3d object categorization. In *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, volume 5, pages 98–107. SCITEPRESS-Science and Technology Publications, 2017. 81

[294] Nabila Zrira, Mohamed Hannat, El Houssine Bouyakhf, and Haris Ahmad Khan. 2d/3d object recognition and categorization approaches for robotic grasping. In *Advances in Soft Computing and Machine Learning in Image Processing*, pages 567–593. Springer, 2018. 63

[295] Nabila Zrira, Haris Ahmad Khan, and El Houssine Bouyakhf. Discriminative deep belief network for indoor environment classification using global visual features. *Cognitive Computation*, 10(3):437–453, 2018. 112

[296] Nabila Zrira, Kawthar Mouhcine, Ibtissam Benmiloud, and El Houssine Bouyakhf. Dynamic texture-based scene classification using deep belief networks. In *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*, page 57. ACM, 2018. 194

[297] Nabila Zrira, Fatima Zahra Ouadiay, Mohamed Hannat, El Houssine Bouyakhf, and Mohamed Majid Himmi. Evaluation of pcl's descriptors for 3d object recognition in cluttered scene. In *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, page 81. ACM, 2017. 42

[298] Nabila Zrira, Younès Raoui, El Houssine Bouyakhf, and Wassima Aitfares. Topological navigation for mobile robot in indoor environment using global visual features. 2014. 159

[299] Zhen Zuo, Gang Wang, Bing Shuai, Lifan Zhao, Qingxiong Yang, and Xudong Jiang. Learning discriminative and shareable features for scene classification. In *Computer Vision–ECCV 2014*, pages 552–568. Springer, 2014. 106

ROYAUME DU MAROC
Université Mohammed V - RABAT - جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

Résumé:

Cette thèse présente une application directe des architectures d'apprentissage profond dans différentes tâches de la robotique. Elle se subdivise en trois parties principales. Dans la partie classification d'objets, nous proposons plusieurs approches qui utilisent des descripteurs 2D/3D ainsi que des réseaux de croyance profonds. Dans un premier temps, nous évaluons les descripteurs les plus existants de la bibliothèque de nuages de points en proposant un nouveau pipeline de la reconnaissance des nuages de points 3D. Deuxièmement, nous proposons de nombreuses approches locales et globales pour classer les objets 2D et 3D à l'aide du sac de mots 2D/3D ainsi que notre nouveau descripteur global Viewpoint Features Histogram-Color (VFH-Color). Troisièmement, une approche globale pour représenter et apprendre des catégories d'objets 3D à l'aide d'un descripteur global et des réseaux de croyance profonds est proposée.

La deuxième partie aborde la classification des scènes qui comprend deux contributions principales. La première est centrée sur des méthodes biologiquement inspirées pour la représentation et la classification des environnements intérieurs. La deuxième contribution fournit une nouvelle fusion de caractéristiques multimodales pour une classification robuste des scènes intérieures RGBD.

La dernière partie de cette thèse présente nos contributions dans le domaine de la navigation topologique. Nous proposons tout d'abord une nouvelle méthode d'exploration d'un environnement intérieur par un robot mobile autonome, ainsi que la création de cartes topologiques. Deuxièmement, nous étendons notre travail précédent de la navigation topologique en utilisant la convolution et la mémoire à long-court termes (C-LSTM) afin de réaliser la cartographie et la localisation topologiques basées sur la reconnaissance des scènes.

Mots-clés : *Robotique mobile, apprentissage profond, classification d'objets, VFH-Color, classification des scènes, navigation topologique, réseau de croyances profond, réseau de neurones à convolution et mémoire à long-court termes.*

Abstract:

This dissertation presents a direct application of deep learning architectures in different mobile robotic. It is encompassed in three major parts. In the object classification part, we propose several approaches using 2D/3D descriptors and Deep Belief Networks (DBNs). First, we evaluate the most existing Point Cloud Library's (PCL's) descriptors by proposing a new recognition pipeline of 3D point clouds. Second, we propose many local and global approaches for classifying both 2D and 3D objects using 2D/3D Bag of Words (BOWs) as well as our new global descriptor Viewpoint Features Histogram-Color (VFH-Color). Third, a global approach for representing and learning 3D object categories using a global descriptor and DBN architectures is proposed.

The second part of this dissertation tackles the scene classification including two main contributions. The first one is centered on biologically inspired methods for representation and classification of indoor environments. The second contribution provides a new multimodal feature fusion for robust RGBD indoor scene classification.

The last part presents our contributions in topological navigation field. First, we propose a new method of exploring indoor environments by an autonomous mobile robot, as well as building topological maps. Second, we extend our previous work of topological navigation by using Convolution Long Short-Term Memory (C-LSTM) in order to perform scene recognition-based topological mapping and localization.

Keywords: *Mobile robotic, deep learning, object classification, VFH-Color, scene classification, topological navigation, Deep Belief Network, Convolutional Neural Network, and Long Short-Term Memory.*

Année Universitaire : 2018/2019