Bootstrapping Robotic Ecological Perception with Exploration and Interactions

Léni Le Goff

Thèse de Doctorat
de Sorbonne Université

Spécialité : **Informatique (EDITE)**

Présentée par : **M. Léni Kenneth Le Goff**

Pour obtenir le grade de
Docteur de Sorbonne Université

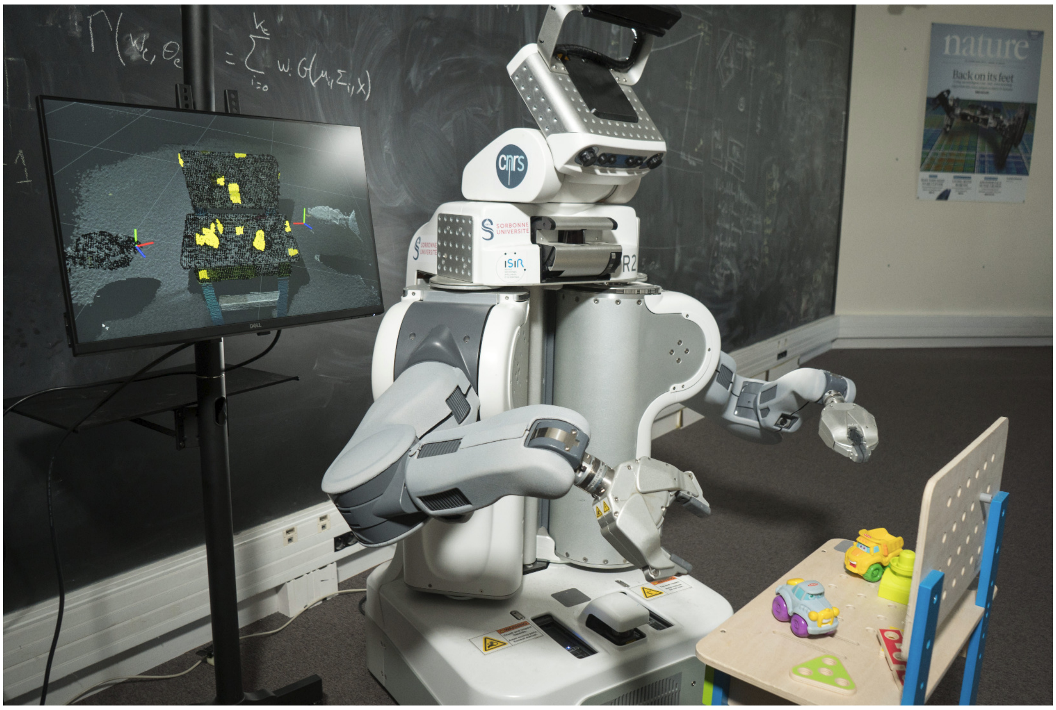
Bootstrapping Robotic Ecological Perception with Exploration and Interactions

Thèse dirigée par **Stéphane Doncieux**

soutenue le 14 Mars 2019

Jury :

Rapporteurs	Pierre-Yves Oudeyer	-	INRIA Bordeaux Sud-Ouest
	Lola Cañamero	-	University of Hertfordshire
Examinateurs	Raja Chatila	-	Sorbonne Université - ISIR
	Franck Guerin	-	University of Aberdeen
	Emre Uğur	-	Bogazici University
Directeur	Stéphane Doncieux	-	Sorbonne Université - ISIR


nature
Back on its feet
cnrs
SORBONNE UNIVERSITÉ
ISIR
PR2

"Résoudre un mystère n'est pas la même chose qu'une déduction à partir de principes premiers. Et ça n'équivaut pas non plus à recueillir une bonne quantité de données particulières pour en inférer ensuite une loi générale. Cela signifie plutôt se trouver face d'une, ou deux, ou trois données particulières qui apparemment n'ont rien en commun, et chercher à imaginer si elles peuvent être autant de cas d'une loi générale que tu ne connais pas encore, et qui n'a peut-être jamais été énoncée."

Umberto Eco, **Le Nom de la Rose**,
Traduit de l'italien par Jean-Noël Schifano.

à Papa et Maman
à Marie-José et Bernard

English Abstract

Robotics has reached a high accuracy on many tasks, like for instance manipulation or navigation. But most of the studies are based on a deep analysis of the problem to solve by the robot designer. These approaches are thus limited to the environment considered by the robot designer, i.e. to a closed environment. Robotics research community is now addressing the issue to allow robots to autonomously achieve tasks in realistic open environments. Such environments are complex and dynamic, like for instance human everyday environment which seems simple but vary a lot from one place to another. In this kind of contexts, the robots must be able to adapt to new situations which were not forecasted by the engineers who designed the robot. Our research work is focused on the development of an adaptive ecological perception for a robotic system. An agent ecological perception defines how it perceives the real world environment through its sensing and acting capabilities. According to J.J. Gibson who has initiated ecological psychology, humans and animals perceive the world through the actions that they can use. Thus, providing a robotic system with the skill to bootstrap autonomously its perception when facing a new unknown situation, would allow the system to be highly adaptive.

Our goal is to provide the robot with the capacity to learn a first representation of its surrounding which would work on any environment. This would allow the robot to learn new representations from unknown situations. It is proposed to generate this ability through an interactive perception method. Interactive perception methods take advantage from action to build or enhance representations of the world and then exploit these representations to have more accurate actions. This relation between action and perception can be easily formalized thanks to affordances. Affordance is a concept introduced by J.J. Gibson which is a relationship between visual features, agent skills, and possible effects.

The system collects data from an environment by interacting with it thanks to a specific action associated to an expected effect. With these data a probabilistic classifier is trained online to build a perceptual map. This map represents the areas that generate the expected effect when the action is applied. Therefore, the map is called a relevance map. Several relevance maps could be built according to different actions and effects, the sum of these maps represents a rich perception of what the robot can do on its surrounding. We name this final map an affordances map as it allows the robot to perceive the environment through the actions it can use. Our methods was tested on the PR2 robots.

Acknowledgement - Remerciement

I would like to thanks my reviewers Lola Cañamero and Pierre-Yves Oudeyer to have read and assess this manuscript. I want also to thanks the other members of my thesis committee Emre Uğur, Franck Guerin and Raja Chatila to have accepted to attend my thesis defense and assess my work. The works of all these researchers had a strong influence on the work I proposed in this manuscript. So, it was a great honor for me to present my thesis work to them.

I deeply thank Stéphane Doncieux my supervisor during 4 years to have given me the possibility to discover the life of a researcher but also the robotics field. His advices and guidance was always relevant and helpful. Despite of all his growing responsibility as the head of our research team, coordinator of the DREAM project and now joint director of our Laboratory, he was always available to discuss with me or to help me. So, thank you Stéphane for all the great discussions and great moments you offered me.

A project, whichever it is, does not succeed without a collective effort. During this Ph.D., I had the chance to work within an European project called DREAM. I want to thanks all the collaborators of this project for the high quality input they brought. Especially, I want to thank David Filliat for his great advices and to had followed my Ph.D. work.

I thank also the European Unions Horizon 2020 research and innovation program for the grant agreement No 640891 which have funded my thesis. And I thank the French government research program Investissements d'avenir through the Robotex Equipment of Excellence (ANR-10-EQPX-44) which have funded the purchase of the PR2 robot.

Being part of a European project means also working with a team daily. I thank Ghanim Mukhtar without whom no experiment would be achieved and Oussama Yaakoubi to have taken the follow up of Ghanim with a great enthusiasm and to have led the last experiments of this thesis. Without these engineers my work will be none.

Carlos Maestre is sort of my big brother of thesis. He supervised me during my internship within our Lab and introduce me into its social life, then we became friend as co-workers. Thank you a lot Carlos for your kindness, your enthusiasm and all your very interesting view about robotics.

Je voudrais remercier aussi mon jumeau de thèse, Pierre Luce-Vayrac. Nous avons commencé, au même moment, notre stage et notre thèse à l'ISIR sur des sujets très proche. Merci Pierre, d'être toujours en désaccord avec moi, pour nos débats sans fin sur la politique et sur le cinéma. Enfin, je te remercie pour nos nombreuses conversations sur les affordances qui m'ont été très utiles.

Je remercie tout les doctorants de la J01. La J01 est notre bureau au laboratoire. Il y a toujours eu une bonne ambiance entre nous ce qui est précieux pendant les moments difficiles. Je voudrais remercier particulièrement les anciens : Antoine, Erwan, Arthur, Thibault, Guillaume et Nassim qui finissaient leurs thèses quand je suis arrivé à l'ISIR. En les observant et en écoutant leurs conseils j'ai pu éviter de nombreux pièges.

Je remercie mes colocataires "les Semi-Indécents" : Alexandre et Hippolytes. Nous avons été en coloc' pendant les 4 ans qu'a durés ma période à l'ISIR. Et c'était bien cool ! Et merci au "Manoir Moustache" : Loïc, Pierre, Louis, Julien et Victor de m'avoir héberger sur les trois derniers mois de ma thèse.

Je remercie mes sœurs et mon frère : Line, Laura et Luc pour leurs soutient autant explicite qu'implicite. Je voudrais remercier Line d'avoir fait une thèse avant moi. Cela m'a permis de voir certaines des difficultés que l'on peut rencontrer.

Cette thèse n'aurait pas été possible sans mes parents. J'ai la chance d'avoir mon père et ma mère comme soutient depuis le début de mes études. Cette chance n'est pas donné à tout le monde. Ils m'ont poussé vers l'excellence non pas pour un élitisme malsain mais pour me permettre d'avoir la liberté de choisir la vie que je voudrais mener. Leurs conseils m'ont été précieux pour devenir qui je suis maintenant. Merci beaucoup Marie-José et Bernard !

Amorcer la perception écologique d'un robot par exploration et interactions

Résumé en français de cette dissertation.

Contents

Résumé

La robotique a atteint une grande précision sur beaucoup de tâches, comme par exemple la manipulation ou la navigation. Mais la plupart des études sont basées sur une analyse complète du problème à résoudre par un ingénieur en robotique. Ces approches sont ainsi limitées aux environnements traités par l'ingénieur, en d'autres termes, à des environnements contrôlés. Aujourd'hui, la communauté de recherche en robotique adresse la problématique de permettre à des robots de réaliser des tâches de façon autonome dans des environnements réalistes et ouverts. De tel environnements sont complexes et dynamiques, comme par exemple notre environnement de tous les jours qui parait simple et structuré mais qui varie beaucoup d'un endroit à un autre. Dans ce genre de contextes, le robot doit être capable de s'adapter à de nouvelles situations qui n'ont pas pu être prévues par les ingénieurs qui l'ont conçu. Notre travail de recherche se concentre sur le développement d'une perception écologique adaptative pour un système robotique. La perception écologique d'un agent qualifie sa perception du monde à travers ses sens et ses capacités d'action. D'après J.J. Gibson qui a initié la psychologie écologique, les humains et les animaux perçoivent le monde par les actions qu'ils peuvent utiliser. Ainsi, un système robotique équipé de la compétence de réinitialiser de façon autonome sa perception quand il fait face à une nouvelle situation, serait hautement adaptatif.

Notre but est de fournir à un robot la capacité d'apprendre une première représentation de ce qui l'entoure, capacité qui fonctionnerait sur n'importe quel environnement. Cela permettrait au robot d'apprendre de nouvelles représentations à partir de situations inconnues. Il est proposé de générer cette capacité par une méthode de perception interactive. Les méthodes de perception interactive prennent avantage de l'action pour construire ou améliorer la perception du monde. Pour ensuite exploiter ces représentations afin d'avoir des actions plus précises. Cette relation entre action et perception peut être formalisée facilement grâce aux affordances. Une affordance est un concept introduit par J.J. Gibson. C'est une relation entre des caractéristiques visuelles, des compétences de l'agent et des effets possibles.

Le système collecte des données de l'environnement en interagissant avec lui grâce à une action spécifique associée à un effet attendu. Avec ces données, un classifieur probabiliste est entraîné en ligne pour construire une carte de perception. Cette carte représente les zones qui génèrent l'effet attendu quand l'action est appliquée. La carte est appelée une carte de pertinence. Plusieurs cartes de pertinence peuvent être construites en fonction de différentes actions et effets, la somme de ces cartes représente une perception riche de ce que le robot peut faire sur ce qui l'entoure. Nous nommons cette carte finale une carte d'affordances puisqu'elle permet au robot de percevoir l'environnement à travers les actions qu'il peut utiliser. Notre méthode a été testée sur le robot PR2.

Introduction

Au-delà d'un scénario préétabli, construire des robots qui sont capables d'agir et de remplir une mission dans des environnements non-contrôlés et non-structurés reste un défi. Même nos environnements de tous les jours, qui tendent à être hautement structurés, sont difficiles à gérer étant données leurs diversités et variabilités. Pour accomplir une tâche dans de tels environnements, le système robotique a besoin d'une perception robuste et adaptative du monde. L'axe de ce travail est d'initialiser l'apprentissage de la représentation du monde d'un robot, en d'autre termes la *perception écologique* d'un robot.

La vision est une modalité riche qui porte un ensemble dense d'informations reflétant la complexité des environnements réalistes. Pour comprendre une scène visuelle, un robot doit d'abord être capable de se concentrer sur les éléments important de son champ visuel en fonction de sa structure corporelle, de ses capacités et de son but courant. Ses processus de décisions ont besoin de connaissances structurées et de représentations du monde avec une dimensionnalité suffisamment basse.

Ainsi, pour choisir l'action appropriée, il a besoin de simplifier son flux sensoriel. Les environnements peuvent être étudiés par des ingénieurs pour faire correspondre la perception à chaque tâche particulière que le robot doit remplir. Dans ce cas, pour avoir une compréhension de la scène utilisable, des hypothèses doivent être formulées par rapport à sa structure.

Par exemple, avec une tâche de triage d'objets, la surface plane la plus grande peut être segmentée, ainsi, les objets posés sur une table sont facilement isolés par le système. Mais cela restreint le robot à des environnements comportant une table comme support.

Une façon d'éviter ce genre d'hypothèse est de permettre au robot d'explorer ce qui l'entoure par interaction direct. Ainsi, en observant l'effet de son action sur l'environnement, le robot peut acquérir de nouveaux signaux sensoriels et apprendre des régularités dans son espace sensorimoteur[1].

Les psychologues E. Gibson et K. O'Regan affirment qu'apprendre à percevoir correspond à l'identification de régularités dans le flux sensorimoteur (Gibson [2000], O'Regan and Noë [2001]). Ces régularités sont des représentations de bas niveau qui permettent la simplification de la vision. De ces régularités, des représentations de plus haut niveaux peuvent être construites comme par exemple des modèles d'objets pour la reconnaissance et la manipulation.

Un domaine de recherche en robotique, connu comme la *perception interactive*, s'intéresse à construire ce genre de représentations en associant perception et interaction (Bohg et al. [2017]). Ce domaine tente d'utiliser l'action pour améliorer la perception et utiliser la perception pour améliorer l'interaction. Néanmoins, les études en perception interactive sont en majorité axées sur la segmentation, la reconnaissance ou la manipulation d'objets. Cela implique de construire un modèle des objets, ce qui est souvent complexe et ce qui contraint le système à partir d'hypothèses soit sur la structure de l'environnement, soit sur les objets eux-mêmes.

L'approche de cette thèse consiste à essayer d'apprendre une représentation de plus bas niveau qui n'implique pas le concept d'objets. De cette façon, moins d'hypothèses spécifiques au contexte ont besoin d'être formulées.

Pour un agent artificiel, apprendre à percevoir en interagissant avec ce qui l'entoure mène à l'acquisition de représentations mettant en relation sa structure physique, ses compétences et son système de vision. Ces représentations sont spécifiques à la structure de son corps et à ses capacités et sont donc mieux adaptées aux caractéristiques du robot. Elles sont aussi construites avec un minimum d'hypothèses, ce qui autorise le robot à faire face une grande diversité de situations. Si un environnement ou une partie d'environnement n'est pas reconnu par le robot, il sera capable de construire une nouvelle représentation. Ces représentations sont des propriétés relationnelles du système agent-environnement ce qui correspond à la définition de l'*affordance*.

Ce concept à été introduit par le psychologue écologiste J.J. Gibson [1966]. Dans ses travaux, il soutenait que les humains et les animaux perçoivent le monde à travers les actions qu'ils peuvent utiliser sur l'environnement.

De plus, pour gérer la complexité, il a été montré que les humains ne considèrent pas toutes les parties d'une scène visuelle comme équivalentes. Les humains possèdent un processus d'attention visuelle qui économise la consommation d'énergie pendant l'analyse d'une scène (Carrasco [2011]). Les éléments d'un environnement

[1] Sensorimoteur qualifie ce qui est relatif à la fois au flux sensoriel et aux informations moteurs

qui sont considérés comme plus saillant attirent l'attention de l'humain (Itti and Koch [2001]).

En traitement de l'image, l'étude de la "saillance" visuelle dans le processus d'attention humaine a mené à la *détection de l'objet le plus saillant* (salient object detection) dans une image (Borji et al. [2014]). Les travaux dans ce domaine ont pour but de produire une *carte de "saillance"* (saliency map) d'images. Cette carte représente une segmentation précise d'un objet dans une image. Néanmoins, la plupart de ces travaux reposent sur de fortes hypothèses et des connaissances a priori pour correspondre à la perception humaine. En combinant les travaux précédents de ce domaine avec la perception interactive, ce genre d'hypothèses peut être évité.

Ainsi avec une carte de "saillance", un robot pourrait directement concentrer son attention sur les éléments importants pour sa tâche et par conséquent baisser les temps de calcul. Cela représente un point de départ pour le développement d'une capacité autonome avec laquelle un robot peut gérer des environnements réalistes avant d'avoir une quelconque représentation d'objets.

L'exploration autonome d'un environnement requiert un système pour la diriger et ainsi la rendre efficace. Dans la robotique développementale, le concept de *motivation intrinsèque* est proposé pour répondre à ce problème.

La motivation intrinsèque est une force directrice pour découvrir et apprendre le monde externe sans guide extérieur. Un système artificiel motivé intrinsèquement n'a besoin d'aucun but pour chercher des nouvelles connaissances. Une motivation intrinsèque peut-être équivalent à un processus d'exploration. La motivation intrinsèque peut être basée, par exemple, sur la maximisation du progrès d'apprentissage comme dans la curiosité artificielle (Oudeyer [2004]), sur la minimisation de l'entropie du modèle ou sur la minimisation de l'incertitude de la représentation (Oudeyer and Kaplan [2009]).

Toutes ces mesures sont extraites du processus d'apprentissage, l'apprentissage *en ligne* est alors plus pratique. En effet, "en ligne" qualifie en robotique tout ce qui se passe lorsque le robot est allumé et agit. Un apprentissage en ligne est donc effectué pendant que le robot exécute sa tâche, comme par exemple un processus d'exploration. L'apprentissage "en ligne" est alors opposé à l'apprentissage "hors ligne" qui s'effectue quand le robot est inactif.

Un domaine de l'apprentissage automatique appelé *apprentissage actif* fournit un cadre pratique pour intégrer à la fois l'apprentissage en ligne et l'apprentissage supervisé avec un processus d'exploration (Settles [2012]). L'apprentissage actif est une famille d'algorithmes qui peut faire la demande de nouveaux exemples pour son entraînement. Les exemples sont choisis pour améliorer au maximum l'apprentissage. La méthode pour émettre la demande d'un exemple est nommée une *stratégie de requête*.

Dans cette thèse, nous proposons un système pour construire une carte de perception similaire à la carte de "saillance" grâce à une exploration autonome mettant en jeu l'action d'un robot équipé de deux bras. Nous nommons cette carte une *carte de pertinence* car elle montre les zones pertinentes en rapport avec l'action utilisée durant l'exploration. Une zone pertinente est une partie de l'environnement qui devrait produire le plus probablement un effet après qu'une action particulière a été appliquée. Ainsi, une carte de pertinence est relative à une action et à un effet associé, en d'autres termes, à une affordance. En combinant différentes cartes de pertinence, une nouvelle carte de perception est obtenue. Nous l'appelons une *carte d'affordances* car elle permet au robot de percevoir son environnement à travers ses actions possibles.

Pour construire une carte de pertinence, un classifieur est entraîné pour discriminer les caractéristiques des éléments qui produisent un effet et ceux qui n'en produisent pas. Le système robotique collecte des échantillons en interagissant avec l'environnement à l'aide d'une primitive d'action. Puis, grâce à un détecteur d'effet, il étiquette ces échantillons. L'entraînement est en ligne, ce qui permet au classifieur de guider l'exploration grâce à une stratégie de requête basée sur la réduction de l'incertitude. Pour apprendre à partir des données collectées pendant l'exploration, un nouvel algorithme d'apprentissage automatique est introduit : *Collaborative Mixture Models* (CMMs, les modèles de mélange collaborateurs). CMMs est basée sur les modèles de mélange Gaussien avec un nombre inconnu de composantes.

Les études décrites dans cette dissertation vise à répondre à la question suivante: *Comment un système robotique équipé de deux bras peut-il, de façon autonome, initialiser sa perception visuelle de l'environnement externe ?*

Cela a mené à trois contributions principales :

- Un processus de segmentation pour identifier les composantes pertinentes en utilisant le paradigme de la perception interactive avec un minimum de connaissances a priori à propos de la structure de l'environnement.
- Un système modulaire d'apprentissage d'affordance pour deux bras robotiques basé sur le paradigme de perception interactive.
- Un classifieur en ligne basé sur les modèles de mélange Gaussien avec un nombre inconnu de composantes et avec une réduction de l'incertitude comme stratégie de requête.

Dans les études présentées dans cette dissertation, le concept d'objets n'est pas considéré. Les problèmes comme l'extraction d'objets pour la reconnaissance ou la manipulation ne sont pas étudiés.

Notre travail constitue plutôt une étape préliminaire permettant une identification initiale avant des explorations plus pointues. Néanmoins, utiliser la méthode proposée comme phase d'initialisation contraindra le genre d'objets qui pourrait être détecté et identifié par les prochaines phases d'apprentissage. Ainsi, nous ne pouvons pas éviter de définir le concept d'objets.

Dans cette thèse, un objet est défini comme une instance physique d'une affordance, c'est-à-dire un objet n'existe ou n'est percevable seulement s'il propose une action pertinente à l'agent. Ainsi, un objet est relatif à la structure corporelle et aux compétences de l'agent. Par exemple, une chaise est nommée comme cela car elle propose l'action de s'asseoir. Pour un insecte, une chaise est à peine un objet, elle est juste un élément du décors. Par contre, une chaise est un objet pour un humain parce qu'il peut s'asseoir dessus. De plus, nous considérons les objets comme une représentation de haut niveau. Dans cette thèse, nous sommes intéressé par le processus initial ascendant qui permet d'analyser efficacement une scène visuelle, alors que les objets sont en général associés à un processus descendant dans lequel les objets reçoivent leurs attributs par rapport à l'agent et sa tâche courante (Chalmers et al. [1992], James [1890]).

Le contexte

Cette thèse s'inscrit dans le cadre de la *robotique développementale*. Ce domaine de la robotique s'inspire de la psychologie développementale laquelle étudie le développement de l'intelligence et de la cognition des enfants humains. L'objectif est de constuire un système robotique avec des compétences adaptatives et avec une compréhension flexible du monde qui est une capacité naturelle chez l'humain.

La robotique développementale est pluridisciplinaire. Elle inclue les collaborations et échanges entre psychologues, neuroscientifiques, linguistes, informaticiens et bien sûr roboticiens. De l'étude de l'intelligence humaine par la neuroscience, la psychologie et les sciences cognitives, de nouvelles pistes de recherches sont identifiées pour développer des systèmes cognitifs artificiels capables de comprendre le monde réel. De plus, tester des hypothèses venant de ces domaines sur des agents artificiels peut mener à d'intéressantes conclusions lesquelles peuvent ouvrir de nouvelles pistes en psychologie et en neuroscience (Cangelosi et al. [2015]).

La théorie de J. Piaget est certainement une référence fondatrice pour la robotique développementale. Il décompose le développement de l'esprit de l'enfant en étapes. Dans une étape, l'enfant construit des représentations, appelées schémas, de lui-même et du monde grâce à son interaction avec lui-même et ce qui l'entoure. Chaque étape de développement repose sur les connaissances construites pendant les étapes précédentes. La première étape repose sur des habilités innées. Piaget propose de diviser le développement de l'enfant en quatre étapes principales :

Première Étape : De zero à deux ans, l'enfant acquiert des schémas sensorimoteurs;

Seconde Étape : De deux à sept ans, des représentations égocentriques et symboliques des objets et actions sont développées;

Troisième Étape : De sept à onze ans, l'enfant peut adopter la perspective des autres personnes sur la représentations d'objets, et ainsi, il peut appliquer des transformations mentales et abstraites sur les objets;

Quatrième Étape : À partir de onze ans et après, cette dernière étape correspond à l'acquisition de pensées abstraites et de résolution de problème complexes.

Par analogie avec la théorie de Piaget, le travail de cette thèse propose un processus d'apprentissage inclus dans la première étape car l'objectif est de construire une première représentation de l'environnement.

Le travail de cette thèse respecte un principe fondamental en robotique dévelopementale proposé par Sutton [2001] : *le principe de vérification.* Ce principe déclare qu'un agent artificiel ne peut créer et maintenir des connaissances sans pouvoir les vérifier lui même. De ce principe découle le fait que, pour un agent, développer une perception ne peut se faire sans des interactions avec l'environnement. En effet, un agent a besoin d'être sûr des signaux senorimoteurs dont il fait l'expérience. Par exemple, pour construire une représentation robuste de la dynamique d'une balle qui roule, le robot doit en faire l'expérience plusieurs fois. Donc, le robot doit pouvoir faire rouler la balle par lui même de manière à ce qu'il puisse faire l'expérience du phénomène à volonté. Par conséquent, la représentation du monde de l'agent artificiel est construit sur des paires acte-conséquence ou sur des actions puis des observations (Stoytchev [2009]).

Cette thèse fait partie du projet européen appelé *Deferred Restructuring of Experience in Autonomous Machine* (DREAM, restructuration différée d'expériences dans les machines autonomes). Dans une approche dévelopementale, le projet DREAM vise à construire des robots qui peuvent s'adapter à leurs environnements et à la tâche à laquelle ils font face. Pour cela l'architecture cognitive du robot inclue des phases de redescription de représentations. Ces représentations peuvent être des modèles de perception, des politiques ou des modèles du monde.

Travaux et domaines proches

Les travaux reportés dans cette dissertation sont liés à différents domaines de l'informatique, de la robotique et du traitement de l'image.

La perception interactive est certainement le domaine le plus proche de des travaux présentés dans cette thèse. Cette dénomination a été introduit par Bohg et al. [2017]. Ils définissent la perception interactive comme l'utilisation de l'action pour améliorer la perception et vice-versa. Le robot, par des interactions avec ce qui l'entoure, révèle de nouveau signaux normalement cachés lors d'une observation passive. De plus, l'interaction permet au robot d'apprendre des régularités dans l'espace combiné des informations sensorielles, des actions, et du temps. Ces régularités sont des associations entre des actions et des signaux sensoriel: quand la même action est exécutée plusieurs fois, les mêmes changements se produisent. Finalement, la perception interactive permet à l'agent d'apprendre la relation de causalité entre les actions et les réponses sensorielles.

La plupart des travaux de ce domaine commence avec une étape de traitement passif de l'image pendant laquelle une première segmentation est faite. Cette segmentation peut être une sursegmentation avec des segments qui sont plus petits que des objets. Dans ce cas, des hypothèses sont utilisées pour maximiser la probabilité d'interagir avec des segments qui font partie des objets. Dans d'autres approches, les segments sont des objets candidats. Les objets candidats sont des groupes de pixels qui représentent soit plusieurs objets, soit une partie d'un objet. Ce sont des zones qui peuvent potentiellement être des objets. Les actions du robots sont faites pour vérifier ces hypothèses.

En général, les approches de ce domaine visent, à la fois, à découvrir et à séparer les objets. Découvrir des objets de façon autonome est une tâche complexes en robotique. L'étape de segmentation initial utilise du traitement de l'image passif. Cette étape requiert une définition préalable de ce qu'est un objet et une formulation d'hypothèses sur la structure de l'environnement.

Les scénarios étudiés sont typiquement restreints à des environnement avec des tables comme support principal sur lesquelles les objets sont posés, ce qui permet de les isoler facilement du décors. Cela introduit une limitation significative sur les types d'environnements qui peuvent être gérés par le robot. Une alternative est de faire des hypothèses sur les objets, comme par exemple sur leurs formes ou leurs textures, mais cela requiert aussi une définition a priori des formes et apparences possibles de tous les objets avec lesquels le robot pourrait interagir.

Le système visuel de l'humain est très efficace pour analyser des scènes complexes. Une des caractéristiques qui rend notre perception visuelle efficace est la capacité de se concentrer sur les éléments saillant dans une scène visuelle. La "saillance" qualifie pour l'humain, ce qui attire son attention ou regard. En général, les objets intéressants sont ceux avec des caractéristiques saillantes.

En traitement de l'image, une famille de méthodes, appelé détection de l'objet le plus saillant (salient object detection), vise à déterminer automatiquement l'élément le plus saillant dans une image (Borji et al. [2014]). Le résultat de telles méthodes est une carte de "saillance" dans laquelle les pixels sont étiquetés comme saillants ou non-saillants.

Les travaux dans ce domaine vise à simuler le système visuel humain grâce à des techniques de traitement de l'image et d'apprentissage automatique. Les méthodes de ce domaine utilisent des hypothèses forte sur ce qui rend un objet saillant pour un humain dans une image.

Un objet sera considéré généralement au centre de l'image, certaines couleurs ou certains types d'objets comme les voitures ou les visages attirent plus le regard humain.

Donc ces méthodes sont utilisées pour construire des modèles de ce que l'humain considère comme saillant. De plus, ces méthodes ne s'appliquent que sur des images statiques, plutôt que sur un flux d'images qu'un robot pourrait collecter lors d'interactions avec l'environnement. L'axe de recherche ici n'est pas de construire une estimation de l'attention humaine. Ici, nous proposons une nouvelle méthode pour détecter les objets pertinents pour le robot en s'inspirant des travaux sur les cartes de "saillances".

Apprendre une représentation visuelle à l'aide d'intéractions avec l'environnement crée un lien direct avec les affordances. Une affordance, un concept introduit par J.J. Gibson, est une propriété relationnelle du système agent-environnement. Habituellement, en robotique, ce concept est défini comme une relation entre un ensemble de sensation, une action et un effet. Ce concept est adapter pour formaliser une représentation de l'environnement en fonction de la structure interne du robot et de ces capacités.

Le concept d'affordance vient du domaine de la psychologie écologique. Cette psychologie, fondé par J.J. Gibson [1966, 1979], étudie la perception humaine au travers de sa structure corporelle et de ses possibilités d'actions. Les psychologues écologistes ont travaillé à préciser la définition donnée par Gibson de l'affordance au départ assez imprécise.

Plusieurs conclusions ont émergé de ces discussions. Les affordances *émergent* de la relation entre l'agent et l'environnement (Chemero [2003]). Un agent peut percevoir l'affordance d'un objet seulement si celui-ci a la connaissance de sa *fonctionnalité* et si il est capable de s'en servir. Enfin, les affordances ne sont pas toujours faciles à "voir". Ainsi, de l'apprentissage et de l'exploration peut être nécessaire pour percevoir celles-ci (Steedman [2002a,b]). Des *symboles* ou *indications* peuvent être construits pour aider un agent à percevoir les affordances. C'est d'ailleurs le travail principal d'un designer lorsqu'il conçoit un objet (Norman [2013]). La fonctionnalité de cet objet doit être facilement *"découvrable"* pour que les actions à effectuer pour s'en servir soit rapidement percevables.

En robotique, une affordance est formalisée soit comme une correspondance entre les signaux sensoriel et les commandes motrices, soit comme un modèle prédictif qui, à partir, d'une entrée sensorimotrice prédit le prochain état sensorimoteur. Dans les deux cas, une affordance est une simplification de l'espace sensorimoteur qui repose sur l'identification d'invariance. Dans le premier cas, elle est en général représentée par une fonction qui représente la correspondance entre la commande motrice et le signal sensoriel (O'Regan and Noë [2001]). Dans le second cas, elle est représentée par une fonction prenant en entrée un état sensoriel et une commande motrice et donnant en sortie un état sensoriel (Krüger et al. [2011a]).

La carte de pertinence

Le but de notre méthode est de produire une carte de pertinence grâce à l'exploration autonome menée par un bras robotique. La carte de pertinence est une segmentation qui sépare ce qui est déplaçable avec une primitive de pousser de ce qui ne l'est pas. Cette segmentation permet au robot de se concentrer sur les parties pertinentes de l'environnement pendant des étapes d'apprentissage ultérieurs ou lorsqu'il résout une tâche. En effet, notre approche pourrait initialiser la plupart des méthodes de perception interactive.

L'exploration consiste en une collecte de données caractérisant ce qui peut être déplacer ou ce qui ne le peux pas. Cette exploration est guidée par la carte de pertinence de l'environnement, qui est construite en ligne. Nous suivons les grands principes des méthodes en perception interactive et en détection de l'objet le plus saillant. Le système sursegmente d'abord la scène en régions et ensuite les classifie pour générer une carte représentant la pertinence des régions. Puis, le système robotique choisit une zone à explorer, interagit avec elle, observe l'effet sur l'environnement, et met à jour le classifieur et la carte de pertinence. La perception repose sur une caméra couleur et de profondeur (Microsoft Kinect 2) qui permet de produire un nuage de points tridimensionnel de la scène.

La première étape est la sursegmentation du nuage de points en régions de taille comparable, appelées *supervoxel* ([Papon et al., 2013a]). Un supervoxel est un groupement de voxels. Un voxel est la plus petite unité visuelle d'une image tridimensionnelle (comme un pixel pour une image). De ces supervoxels, des caractéristiques visuelles sont extraites. Ces caractéristiques visuelles sont données en entrée au classifieur. Une caractéristique visuelle caractérise un supervoxel. La caractéristique utilisée est la concaténation d'histogrammes de couleur et d'un histogramme géométrique (fast point features histogram Rusu et al. [2009]).

Ensuite, la carte de pertinence est calculée sur la base de la sursegmentation et en fonction de la prédiction du classifieur. Chaque supervoxel reçoit un poids d'une valeur entre zero et un. Ces valeurs représentent les pertinences des supervoxels. Ces valeurs de pertinence sont les prédictions du classifieur entrainé en ligne pendant l'exploration. Elles représentent la probabilité d'un supervoxel de faire partie de "quelque chose" de déplaçable par le robot. Le classifieur sera décrit avec plus de précision dans la prochaine section.

Une fois que la carte de pertinence est extraite, le système choisit le prochain supervoxel avec lequel interagir. Ce choix est guidé par une stratégie de requête qui repose sur la carte de pertinence. Une carte de distribution de choix est calculée grâce à la stratégie de requête du classifieur qui représente la probabilité d'une région d'être choisie. Le calcul de probabilité repose sur l'incertitude et la confiance du classifieur et la diversité de la base de données. La stratégie de requête sera décrite plus en détail dans la prochaine section.

Le robot interagit avec le supervoxel sélectionné avec une primitive de pousser. La cible de cette primitive est le centre du supervoxel. Cette primitive est divisée en trois étapes : un mouvement d'approche de la cible, un mouvement en ligne droite vers le centre du supervoxel et enfin un mouvement inverse pour revenir à la position de départ. Avec cette primitive le robot essaye de pousser "quelque chose" dont fait partie le supervoxel.

Pour observer si la primitive a produit un mouvement, un détecteur de changement dans l'image est appliqué. Ce détecteur est simplement une comparaison entre le nuage de points avant l'interaction et le nuage de points après l'interaction. Cette comparaison n'est faite qu'entre les points faisant partie du supervoxel dans les deux nuages de points. Ainsi, la détection de changement n'est appliquée qu'autour de la cible. Finalement, si un changement est détecté, la caractéristique visuelle du supervoxel est stockée dans la base de donnée avec une étiquette égale à un et si rien n'est détecté alors elle est stockée avec une étiquette égale à zéro.

Pour évaluer l'approche, deux séries d'expériences ont été effectuées, l'une en simulation et l'autre sur le robot PR2 sur un environnement réel. Celles en simulation permettent de tester la méthode dans un cas idéal.

En simulation, les expériences ne font pas intervenir de robots, les interactions sont fausses, la classe des supervoxels explorés est donnée par un expert qui connaît à l'avance l'étiquette de chaque supervoxel. Bien entendu le classifieur n'a pas accès à l'expert. Ainsi, durant ces expériences en simulation, il n'y a pas de bruit et d'échantillons mal étiquetés. Les résultats de ces expériences nous permettent d'avoir une borne supérieur de ce qui peut être attendu en réalité. Les expériences sont effectuées sur six environnements différents conçus pour avoir une difficulté croissante.

Les expériences dans le monde réel sont effectuées avec le robot PR2 équipé d'une Kinect v2. Le robot explore un environnement constitué d'un plan de travail pour enfant avec des objets libres ou fixés dessus. Cet environnement est coloré et a des formes complexes ce qui permet de tester la méthode sur une installation réaliste. Dans ces expériences, le robot interagit réellement avec l'environnement et le classifieur apprend des échantillons étiquetés par ces interactions.

Le résultats des expériences en simulations sont très bons sur les environnements faciles et satisfaisants sur les environnements plus complexes. Les expériences avec le robot réel donnent de moins bon résultats mais la qualité reste suffisante pour produire des cartes de pertinence utilisables.

Néanmoins, sur les environnements réelles la qualité du classifieur présente une certaine instabilité ce qui vient probablement d'échantillons mal étiquetés à cause d'une interaction ratée ou d'une erreur du détecteur de changement.

Les modèles de mélange collaborateurs

Comme déjà mentionné, l'approche présentée dans cette thèse utilise la perception interactive pour collecter des données qui sont ensuite traitées par un classifieur. La méthode de classification doit être capable de gérer une grande variété d'environnements.

La méthode doit exhiber les propriétés suivantes : pouvoir gérer des bases de données non-linéairement séparables et des espaces de données non-convexes, avoir une mesure d'incertitude pour la stratégie de requête, avoir des hyperparamètres non-spécifiques à un environnement particulier, être supervisée et adaptée à la classification et enfin pouvoir être entraînée en ligne.

Pour classifier les échantillons collectés, une nouvelle méthode de classification est introduite : les modèles de mélange collaborateurs (CMMs).

L'algorithme suit le protocole de l'apprentissage actif (active learning): les échantillons sont collectés et étiquetés par l'interaction du robot, puis ajoutés à la base de données. Le prochain échantillon est choisi grâce à une stratégie de requête basé sur la réduction de l'incertitude. Pour chacune des classes, les échantillons collectés sont représentés par un ensemble de groupes encodées par une distribution normale multivariée. Elles sont sommées pour former un modèle de mélange. Il y a donc un modèle de mélange Gaussien par classe. Les paramètres de la distribution normale multivariée sont estimés en utilisant la moyenne et la covariance statistique.

Le nombre de composantes de chaque modèle n'est pas donné a priori et est adapté à l'ensemble d'entraînement. Chacun des modèles commence avec une composante et ajoute ou supprime une composante avec des opérations de fusion et de division. Ces opérations adaptent le nombre de composantes aux données. Les distributions normales multivariées sont approchées par des hyper-ellispoïdes de tolérance. L'intersection entre ces ellipsoïdes peut être testée, ce qui permet de tester si deux distributions se chevauchent. Si deux distributions encodant des échantillons de différentes classes s'intersectent, alors l'une d'elles doit être divisée en deux nouvelles distributions. Par contre, si deux distributions encodant des échantillons de la même classe s'intersectent, alors elles doivent être fusionnées.

Le système a le choix de l'échantillon qu'il peut collecter et a donc besoin d'un algorithme capable de donner une indication sur l'échantillon qui sera le meilleur pour améliorer sa classification. Pour cela une stratégie de requête est utilisée pour générer une carte de distribution de choix sur les supervoxels extraits sur la scène courante.

Cette stratégie est basée sur deux mécanismes. Le premier est basé sur l'estimation de l'*incertitude*, ce qui permet au système de se concentrer sur les zones incertaines de l'environnement. Mais aussi cette estimation de l'incertitude pousse le système à explorer la classe avec le moins de d'échantillons collectés. Cela données dans les zones de l'espace des données dans lesquelles peu de données ont été collectées. Autrement dit, les zones dans lesquelles le classifieur à peu d'information.

Les CMMs sont évalué dans toutes les expériences présentées dans cette thèse. Néanmoins, trois séries d'expériences ont été conduites spécialement pour évaluer les différentes composantes du classifieur.

Une première série vise à tester l'intérêt des opérations de fusion et de division en faisant varier l'hyperparamètre α qui contrôle la sensibilité du critère d'intersection de deux distributions normales. Cette série d'expériences est conduite, d'abord, sur la base de données MNIST (LeCun et al. [2010]). C'est une base de données de chiffres manuscrits et comportant donc 10 classes. Puis, elles sont conduites sur des environnements réels filmés par une Kinect v2. Ici, le robot n'est pas utilisé, les étiquettes des échantillons sont données par un expert. L'absence de robot permet de raccourcir significativement la durée des expériences. La seconde série d'expériences vise à comparer différentes caractéristiques visuelles. Et enfin la troisième compare différentes stratégies de requête: aléatoire uniforme, incertitude seule, confiance seule et confiance combinée avec l'incertitude.

À l'issue de ces expériences, l'apport des opérations de fusion et de division ont été démontrée. De plus, la variabilité des résultats quand α varie est relativement basse.

La contribution de l'incertitude et de la confiance dans la stratégie de requête a été démontrée. Néanmoins, la contribution de la confiance semble faible.

La comparaison entre différentes caractéristiques visuelles a mené à choisir une combinaison d'histogrammes de couleur avec l'encodage CIELab[2] et un histogramme géométrique (FPFH).

Enfin, à la suite de ces expériences, un nouvel hyperparamètre est introduit : un nombre maximale de composantes par modèle de mélange gaussien. Ceci pour éviter une explosion du nombre de composantes observée pendant les expériences décrites dans la section précédente.

La carte d'affordances

La méthode présentée permet de construire une carte de pertinence de ce qui peut être déplacé grâce à une primitive de pousser. Sans changer le cœur et la structure de la méthode, des cartes de pertinence relatives à d'autres primitives d'action peuvent être apprises.

Des expériences sont menées avec une primitive de pousser, une primitive de soulèvement et une primitive pour appuyer sur un bouton poussoir. Chaque carte de pertinence produite est alors relative à une affordance différente : la poussabilité, la soulevabilité et l'activabilité de bouton poussoir. Les cartes issues de ces expériences sont alors combinées pour produire une *carte d'affordances*. Ainsi, avec cette carte, le système a une riche perception de son environnement au travers des actions qu'il peut appliquer.

Dans ce travail, une affordance est formalisée comme étant une probabilité conditionnelle d'une donnée visuelle de faire partie d'un objet produisant un effet après l'application d'une action sur celui-ci.

L'approche est similaire à celle décrite précédemment, seulement pour chaque série d'expériences la primitive d'action et le détecteur d'effet changent. En effet, chaque primitive est associée à un effet prédéfini pour être détectable. Les expériences sont effectuées avec le robot PR2 sur un environnement unique comportant des boutons poussoirs, des objets poussables et des objets soulevables par le robots. Cet environnement est une cuisine dînette pour enfant.

Les expériences menées pour évaluer les cartes d'affordances sont des preuves de concept. Les résultats ont montré une grande variabilité pour chacune des affordances et les classifieurs entrainés pour les affordances des éléments poussable et des boutons poussoir activables divergent à la fin des expériences. Néanmoins, de

[2] CIELab est un standard international de colorimétrie décidé pendant la commission international de l'éclairage (CIE) de 1978

ces expériences, des cartes de pertinence appropriées pour être combinées en une carte d'affordances significative ont été produites.

Conclusion

Dans cette thèse, nous avons proposé un cadriciel pour apprendre une carte de perception, appelé carte d'affordances, par l'exploration autonome d'un environnement par des interactions. Une carte d'affordances est la combinaison de plusieurs cartes de pertinence dont chacune est spécifique à une affordance. Le but principal était d'apprendre une représentation avec un minimum d'hypothèses particulières à un environnement. Finalement, l'approche repose sur peu d'hypothèses :

- La primitive d'action est capable de produire des effets pouvant être détectés avec un détecteur d'effet, ou, autrement dit, le détecteur d'effet est capable de détecter les effets produits par une primitive d'action.
- Les plus petites parties de l'environnement avec lesquelles le robot peut interagir sont plus grandes que les supervoxels.
- Le décor est suffisamment différent des éléments qui proposent la primitive d'action pour être séparable par un classifieur.

Ces hypothèses ne sont pas spécifiques à un environnement particulier. Néanmoins la méthode est limitée aux environnements qui sont adaptés aux robots avec des bras et à des tâches de manipulation. Nous n'avons pas, par exemple, considéré de robots mobiles et de tâches de navigation.

Le cadriciel est composé de quatre modules différents : un classifieur, un extracteur des caractéristiques d'un supervoxel, une primitive d'action et un détecteur d'effet. Ces modules sont relativement indépendants et peuvent être changés pour apprendre des représentations relatives à différentes affordances ou accroître la performance.

L'approche proposée ici vise à être une étape d'initialisation pour d'autres étapes de développement. Les méthodes de perception interactive reposent sur une étape d'initialisation qui utilise du traitement de l'image passif pour produire une segmentation représentant, soit des objets candidats soit des parties d'objets. Cette étape introduit des hypothèses sur la structure de l'environnement et des objets, ce qui réduit les capacités d'adaptation du système. Elle pourrait être remplacée par notre approche. Cela supprimerait beaucoup d'hypothèses, et ainsi, rendrait ces approches pertinentes sur une plus grande variété d'environnements.

Dans le projet DREAM, la prochaine étape de développement sera une exploration axée sur les objets en utilisant une carte de pertinence avec le but de collecter des données visuelles de couleurs et profondeurs et des données proprioceptives du robot. De ces données, des modèles d'objets peuvent être appris hors ligne en utilisant par exemple de l'apprentissage profond. Des modèles tridimensionnels des objets peuvent ensuite être utilisés avec des méthodes d'apprentissage requérant de nombreux tests. Ainsi le robot peut apprendre à les manipulés.

Contents

Chapter 1

Introduction

Beyond a preprogrammed scenario, building robots that are able to act and fulfill a mission in uncontrolled and unstructured environments remains a challenge. Even everyday environments, which tend to be highly structured, are hard to deal with given their variability. To achieve tasks in such environments, a robotic system needs a robust and adaptive perception of the world. The focus of this work is to bootstrap the learning of the world's representation of a robot, i.e. a *robotic ecological perception*.

Vision is a rich modality that carries a dense set of information reflecting the complexity of realistic environments. To understand a visual scene, a robot must first be able to focus on important components of its visual field according to its embodiment, skills, and current goal. Its decision processes need structured knowledge and sufficiently low dimensional representations of the world. Therefore, to select an appropriate action, it needs to simplify this sensory flow. Environments can be studied by engineers to fit the perception to specific tasks the robot has to achieve. But, to get a practical scene understanding, hypotheses must be formulated relating to the structure of the studied environment. For instance, with an objects sorting task, the largest plane could be segmented, thus, the objects laying on a table are easily isolated by the system. But this restricts the robot to tabletop environments. In this way, the robotic system is restricted to progress in specific environments.

A way to avoid making such assumptions is to allow the robot to explore its surrounding via direct interaction. In this way, by observing the effect of its action on the environment, the robot can acquire novel sensory signals and learns from regularities in its sensorimotor[1] space. Psychologists E. Gibson and K. O'Regan claim that learning to perceive corresponds to the identification of regularities in the sensorimotor flow (Gibson [2000], O'Regan and Noë [2001]). These regularities are the lowest level representations that permit simplification of vision. From these regularities, higher level representations can be built like for instance objects model for recognition and manipulation. A domain of robotics research, known as *interactive perception*, is interested in building such representations by associating perception and interaction (Bohg et al. [2017]). This domain attempts to use action to enhance perception, and use perception to enhance interaction. However, the studies in interactive perception are mostly focused on object segmentation, recognition or manipulation. This implies to build a model of objects which is often complex and constrains the system to start from assumptions about either the environment

[1] Sensorimotor refer to the coupling of the sensor system and the motor system for an agent

structure or the objects themselves. The approach of this thesis consists in trying to learn lower level representations which do not imply the concept of object. In this way, fewer assumptions specific to a context need to be formulated.

For an artificial agent, learning to perceive by interacting with its surrounding leads to the acquisition of representations which link its embodiment, skills, and visual system together. These representations are specific to its body structure and current capacities and are then more adapted to the robot features. These representations are also built with minimum assumptions which allow the robot to face a large variety of situations. If an environment or part of an environment is not recognized by the robot, it will be able to build a new representation. These representations are relational properties of the agent-environment system, which corresponds to the definition of an *affordance*. This concept was introduced by the ecological psychologist J. J. Gibson [1966]. He argues that humans or animals perceive the world through the actions they can use on the current environment.

Also, to deal with complexity, it has been shown that humans do not consider all parts of a visual scene as equivalent. Humans possess a visual attention process that lowers energy consumption during the analysis of a scene (Carrasco [2011]). Elements of the environment that are considered more salient attract the attention of a human (Itti and Koch [2001]).

In computer vision, the study of visual saliency in human attention has led to salient object detection within an image (Borji et al. [2014]). Work in this field has aimed at producing a *saliency map* of images, i.e. a binary map representing an accurate segmentation of an object in an image. However, most of these works rely on strong assumptions and prior knowledge to fit with human perception. By combining previous work of this field with interactive perception such assumptions can be avoided. Thereby, with a saliency map, a robot could directly focus its attention on elements important for its task and thus decrease computational time. This represents a starting point for developing an autonomous capacity with which a robot can deal with realistic environments prior to having any representation of objects.

Autonomous exploration requires a driving system which allows the robot to explore efficiently its environment. In developmental robotics, the concept of *intrinsic motivation* is proposed as an answer to this issue. Intrinsic motivation is a driving force to discover and learn about the external world without any external guidance. An artificial system intrinsically motivated does not need any goals to seek new knowledge. Intrinsic motivation can be based on maximizing the learning progress as in artificial curiosity (Oudeyer [2004]), minimizing the entropy or minimizing the uncertainty of the representation (Oudeyer and Kaplan [2009]). Thus, these driving systems are based on measures extracted from the learning process itself, online learning is, then, more practical. A field of machine learning called *active learning* (Settles [2012]) provides a suitable framework to integrate online and supervised learning with a sampling process. Active learning is a family of algorithms that can query specific unlabeled samples which would maximize the learning progress. The method to query samples is named a *query strategy*.

In this thesis, we propose a framework to build a perceptual map similar to saliency map by autonomous exploration involving interactions of robotic arms. We name this map a *relevance map* because it shows the relevant areas according to the action used during the exploration. A relevant area is a part of the environment which will most likely produce an effect after a specific action has been applied. Thus, a relevance map is relative to an action and an effect, in other words, to an affordance. By combining several different relevance maps, a new perceptual map is obtained. We call it an *affordances map* as it allows the robot to perceive its environment through its own possible actions.

To build relevance maps, a classifier is trained to discriminate features from elements which produce an effect and those which do not. The robotic system collects samples by interacting with the environment with an *action primitive* and thanks to an *effect detector* labels the collected samples. The training is online, which allows the classifier to drive the exploration thanks to a query strategy based on uncertainty reduction. To learn from data collected during exploration by interaction, a new machine learning algorithm is introduced: *Collaborative Mixture Models* (CMMs). CMMs is based on Gaussian mixture models with an unknown number of components.

The studies described in this dissertation aim at answering the following question :

How can a robotic system equipped with two arms autonomously bootstrap its visual perception of the external environment ?

Which has led to three main contributions :

- A segmentation process to identify relevant components using the interactive perception paradigm with minimum a priori knowledge about the environment structure.
- A modular affordance learning framework for two arms robotic systems based on the interactive perception paradigm.
- An online classifier based on Gaussian mixture models with an unknown number of components with an uncertainty reduction query strategy.

In the studies presented in this dissertation, the concept of object is not considered. Problems such as objects extraction for recognition or manipulation is not addressed. Our work rather constitutes a previous step, allowing an initial identification prior to a more targeted exploration. With minimum a priori knowledge on the environment, this work represents the very first step of a developmental process which could lead to the acquisition of robust and adaptive unknown object extraction and identification. However, using the proposed method as a bootstrap phase will constrain the kind of objects that could be detected and identified in further learning phases. Thus, we cannot avoid defining the concept of object. In this thesis, an object is defined as a physical instance of an affordance, i.e. an object exists or is perceived only if it affords a relevant action for the agent. Thus, an object is relative to the embodiment and to the skills of the agent. For instance, a chair is called like that because it affords the action of sitting. For an insect a chair is barely an object, it is just an element of the background. A chair is an object for a human because he can sit on it. Moreover, we consider objects as high level representations. In this thesis, we are interested in the initial bottom-up process which allows an agent to efficiently analyze a visual scene. While objects are frequently associated with a top-down process in which objects receive their attributes relatively to the agent and its current task (Chalmers et al. [1992], James [1890]).

The rest of the dissertation is organized as follows :

Chapter 2: Introduction of the context in which this thesis fits. First, developmental robotics is summarized, then classification problems and supervised learning are introduced. The chapter ends with a brief description of the related works (more detailed descriptions are in the other chapters).

Chapter 3: Explanation of the technical background necessary to understand our approaches. First, Gaussian mixture models are introduced, then the computer vision methods used in this thesis are described.

Chapter 4: Detailed description of CMMs, the proposed classifier, with, as a preamble, a state of the art of online learning, Gaussian mixture model with an unknown number of components and query strategies used in active learning.

Chapter 5: Presentation of an approach to build a relevance map based on a push primitive. The chapter begins with a review of interactive perception and salient object detection, then the approach is explained and finally experimental results are presented.

Chapter 6: Following the conclusions of chapter 5, an extensive study to evaluate CMMs is presented. Each aspect of the classifier is evaluated experimentally.

Chapter 7: After a review of affordance learning in robotics, an extension of the approach presented in chapter 5 is presented. Three relevance maps based on a push primitive, a lift primitive and a push button primitive are learned to be finally combined in an affordances map.

Chapter 2

Context

Contents

2.1 Introduction

In this chapter, the context and the related works of this thesis are described. First, the field of developmental robotics is presented and its main principles are enumerated and explained. The work of this thesis is designed to be integrated into a developmental cognitive architecture as an initial step to develop an ecological perception for a robot. Then, the classification problem is defined and three families of machine learning algorithms are briefly described: supervised, semi-supervised, and active learning. This last one is used in this thesis to solve classification problems. Finally, the related domains: interactive perception, online learning, Gaussian mixture models, query strategies, saliency map, and affordance learning are summarized and related works which cover most of these domains are described.

2.2 Developmental Robotics

This thesis come within the scope of developmental robotics. Developmental robotics is defined by Cangelosi et al. [2015] in their book "Developmental robotics: From babies to robots" as :

> *"The interdisciplinary approach to the autonomous design of behavioral and cognitive capabilities in artificial agents (robots) that takes direct inspiration from developmental principles and mechanisms observed in the natural cognitive systems of children."*

This field of robotic draws inspiration from developmental psychology which studies the development of intelligence and cognition of human infant. The aim is to build a robotic system with adaptive skills and with a flexible understanding of the world which is a natural capacity of humans. This field is interdisciplinary as it includes collaborations and exchanges between psychologists, neuroscientists, linguists, computer scientists and of course roboticists. By the study of human intelligence through neuroscience, psychology and cognitive science, new research directions are identified for developing artificial cognitive systems able to comprehend the real world. Also, testing assumptions from those fields on artificial agents may lead to interesting conclusions which open new paths in psychology and neuroscience. Developmental robotics takes its origin into two international conferences *International Conference on Development and Learning (ICDL)* and *Epigenetic Robotics (EpiRob).* ICDL was established after the workshop on Development and Learning held in 2000 at Michigan State University. While EpiRob followed the first international workshop on Epigenetic Robotics held in 2001 at Lund University (Sweden). The term "epigenetic" is a reference to the *epigenetic theory* of the psychologist J. Piaget. In 2011, ICDL and EpiRob have merged to become IEEE ICDL-EpiRob and this field is now mainly called Developmental Robotics.

The theory of J. Piaget is certainly a core reference for developmental robotics. He decomposes the development of infant mind into stages (Piaget and Cook [1952]). In a stage, the child builds representations, called *schemas* of himself and of the world thanks to the interactions he has with it. Each developmental stage relies on the knowledge built during previous stages. The first stage relies on innate abilities. Piaget proposed to segment the infant development into four main stages :

Stage 1 From 0 to 2 years old, the child acquires sensorimotor schemas;

Stage 2 From 2 to 7 years old, egocentric symbolic representations of objects and actions are developed;

Stage 3 From 7 to 11 years old, the child can adopt other people's perspective on object representations, and thus, he can apply abstract mental transformations on objects;

Stage 4 From 11 years old and after, this last stage corresponds to the acquisition of abstract thinking and complex problem solving.

By analogy with Piaget's theory, this thesis work proposes a learning process included in the first stage, as it aims at building a first representation of the environment.

In the literature, roboticists have proposed some principles to qualify a work as a developmental robotics one. Cangelosi et al. [2015] proposed 6 main principles or properties of a developmental robotics research work :

- *Development as a dynamical system*: Considering the body as a complex system with stochastic and dynamic aspects.

- *Phylogenetic and ontogenetic interactions*: Ontogenetic corresponds to the maturational changes and phylogenetic to evolutionary changes. Evolutionary robotics, a subfield of developmental robotics studies, is based on phylogenetic interactions. They use evolutionary algorithms to build robotics controllers (Doncieux et al. [2015], Cully et al. [2015], Lehman et al. [2018], Lehman and Stanley [2011]).

- Embodied and situated development state the necessity to have a body to learn and to interact with the environment in a certain context. This is the main claim of interactive perception studies Bohg et al. [2017].

- Intrinsic motivation, like curiosity, novelty, or surprise (Oudeyer [2004], Oudeyer and Kaplan [2009]) and social learning which states the importance to learn with others thanks to imitation and demonstration.

- Nonlinear, stage-like development: The development of an artificial agent must be organized into step like suggests by Piaget's theory.

- Online, open-ended, cumulative learning: The robotic system should be able to learn continuously during its "life" by accumulating knowledge.

A cognitive architecture for developmental robotics should gather these properties. The work presented in this thesis does not exhibit all of these properties as it is a brick in a bigger developmental framework.

Stoytchev [2009] has proposed a list of principles for the development of artificial intelligence (AI) which can apply to most works in developmental robotics. They all come out of the *Verification Principle*, defined by Sutton [2001] as follows :

> *"An AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself."*

This principle is crucial for developmental robotics. To build an artificial agent able to learn from the real world autonomously, it has to be able to assess the acquired knowledge, representations or abstractions alone. Stoytchev [2009] first extracts from it *the principle of embodiment* which states that an AI needs a body to interact with the real world. This principle follows directly the verification principle because, without the possibility of acting, an agent would not be able to verify anything by itself. If a robot can learn only from its own actions and from the feedback of its sensors, its representation will be specific from its embodiment. This is *the principle of subjectivity.* In other words, the principle of subjectivity states that an AI can learn only from its own experiences. This implies, on one hand, the notion of sensorimotor limitations: the robot can only learn what it can experience according to its body and sensors capability. On another hand; it results in the notion of experiential limitations: the knowledge the robot can build is limited by its own experience. This means in particular that the robot acquires experiences progressively and thus does not learn everything at the same time. Some representations need other more basic representations to be built, thus, the learning must be incremental from simple knowledge to more complex abstractions. This is *the principle of incremental development* which meets the Piaget's theory. Finally, The *the grounding principle* states that all knowledge must be grounded somehow. The agent needs to be sure of the sensorimotor signals it experiences. For instance, to build a robust representation of the dynamic of a rolling ball, the robot must experience it several times. So, the robot must be able to make the ball roll by itself so that it can experience the phenomenon as much as it wants. Therefore, the world's representations of an artificial agent are built on *act-outcome* pairs or from actions then observations.

The study presented in this dissertation is based on all above-mentioned principles. Thus, it is based mainly on the verification principles.

This thesis is part of a European project called *Deferred Restructuring of Experience in Autonomous Machines (DREAM).* The DREAM project aims at building robots that can adapt to their environment and to the tasks they face, through a developmental approach focused on the redescription of the representations the robot cognitive architecture needs; including perceptions, policies and world models (Doncieux et al. [2018]). This work fits in the bootstrap phase of this process in which the robot needs first to identify the structure of the environment to collect relevant data from interaction before learning relevant state spaces (Raffin et al. [2018]) and motors skills (Kim et al. [2019]) that can be later on redescribed for a better generalization (Jegorova et al. [2018]).

2.3 Classification problem and learning methods

2.3.1 Classification Problem

Given a continuous space F and a discrete set of value L, a classification problem requires to find a function $h : F \rightarrow L$. F is generally called the feature space and L a set of labels which correspond to classes. The mapping function h between F and L is, in general, based on a dataset $D = (x_i; y_i)_{i \in K}$ of examples where $x_i \in F$ and $y_i \in L$. The goal is to categorize a newly incoming data from F into one of the K classes based on previously seen examples (Marsland [2011]). Three main assumptions hold in a classification problem: the classes are discrete, e.i., L is discrete space, each sample belongs to only one class, and the feature space is totally covered by the classes. Using machine learning to solve such a problem is relevant if all examples cannot be seen, which is the case as F is continuous. In this case, the main feature of a classifier is its generalization ability.

Theoretically, a learning algorithm must find the right parameters θ of a model h to map all the samples of the training dataset D with their labels and also minimizes the *generalization error* by reducing the error of classification on the training dataset. This definition is formalized in Equation 2.1.

$$
\begin{aligned}
D &= (x_i; y_i)_{i \in K} \\
\theta^* &= argmin_\theta(\sum_i^K y_i - h_\theta(x_i))
\end{aligned}
\tag{2.1}
$$

Where D is the training dataset with x_i an example and y_i its true label; h_θ a model with parameters θ;

This formulation of classification problem corresponds directly to *supervised learning*: The learning algorithm processes a training dataset to estimate the parameters by minimizing the classification error. The optimization strategy used is specific to the type of model. The underlying assumption in supervised learning is that the training datasets are representative of the problem, i.e. processing of this dataset is sufficient for a generalization to the rest of the feature space. A test dataset can be used to compute an approximation of the generalization error and thus estimate the quality of the training. The test dataset must be sufficiently different from the training dataset.

2.3.2 Semi-supervised and Active learning

In *semi-supervised learning* the assumption of a representative training dataset does not hold as this family of methods is interested in incomplete data classification problems. The training algorithm uses, in addition to labeled data, unlabeled data to enhance the classification. This is useful when the dataset available is small or if the labeling process is costly. Semi-supervised learning is a mix of supervised learning and unsupervised learning techniques. Training algorithms infer or predict the missing labels by data analysis and clustering. In Hady and Schwenker [2013], they distinguish 4 kinds of semi-supervised learning methods: with generative models, with support vector machines, with graph, and with committees. The semi-supervised learning with generative models uses a probabilistic representation with mixture models. First, a distribution is estimated with the labeled dataset then unlabeled data are processed with expectation-maximization. This family of methods is the closest to the classifier presented in this thesis (see section 4 about CMMs). For image processing, the Gaussian mixture models are the most frequently used. Semi-supervised support vector machines estimate the decision boundary by getting through low-density region using unlabeled data and by respecting the labeled data. In this way, labels are assigned to the unlabeled data. The main drawback of these methods is the complexity of the optimization problem which combines the classic support vector machine and the labeling of unlabeled data. Semi-supervised learning with graphs consists in constructing a graph with labeled and unlabeled data based on a similarity metric. A minimum cut of the graph is applied to classify the samples. Finally, semi-supervised learning with committees is based on ensemble learning. An ensemble of diverse classifiers is trained either on different feature spaces representing the same data or the same feature space. The diversity of the classifiers is very important, thus a measure of disagreement is required. The algorithm is called co-training. After a training in small sets of labeled data, each classifier predicts the label of unlabeled samples and the samples are ranked by confidences of the classifiers. Then the most confident example is added to the training dataset.

This last family of semi-supervised learning is very close to *active learning*. Active learning considers also incomplete dataset and aims at learning from small training datasets. In active learning, two sets of data are considered, one with labeled data and one with unlabeled data. Unlike semi-supervised learning, algorithms of active learning do not exploit unlabeled dataset with unsupervised methods but instead, the training algorithm queries data from the unlabeled set which will be the most useful to enhance the representativeness of the dataset. Then, the chosen data is labeled by an oracle (most of the time a human annotator). The underlying assumption is that the training will need less training data to have better performance than a random choice of the data processed like in batch learning. Section 4.4 goes into more details about active learning.

The proposed framework in this thesis is very closed from active learning. But instead of a human, it uses a robot interacting with the environment as an oracle to collect data.

2.4 Related Works

2.4.1 Related Domains

The work reported in this dissertation is related to different fields of computer science, robotics and image processing. The following section summarizes the closely related works in each of these domains. A deeper description of related works is included in each specific chapter.

Interactive Perception

The closest field to my work is certainly *interactive perception*. This expression was introduced by Bohg et al. [2017]. They define interactive perception as the use of action to enhance perception and, then, a better perception to have more accurate actions. The robot, through its interactions with its surrounding, reveals novel sensory signals normally hidden in a passive observation. These sensory signals can be exploited to learn a representation of the world that links the action and perception. This field is described in details in section 5.2.

Interactive perception is used as a framework for the work presented in chapter 5 which aims to build a relevance map.

Online Learning, Query Strategies, and Gaussian Mixture Models

The approach presented in this thesis uses interactive perception to collect data which are processed by a classifier. Therefore, the system has the choice of the samples it can collect and needs an algorithm able to produce an insight of what would be the next best sample to enhance its classification. In active learning (see the previous section) this is called a query strategy. Different query strategies proposed in the literature are described in 4.4. In this context, the samples arrive one by one, thus, online learning could be used. Online learning is a family of machine learning algorithms in which the data are processed on the arrivals. This is opposed to offline learning in which a complete dataset is processed by the algorithms. The dataset could be processed several times. Online learning methods are described in section 4.2.

Finally, learning from real data introduces uncertainty and variability, thus, statistical machine learning models such as Gaussian mixture models (GMM) are practical. GMMs are known for being regression methods able to approximate a large variety of distributions. A GMM is a weighted sum of normal distributions called a component. GMMs are defined in section 3.2, then, mixture models trained online are described in section 4.2. GMMs are generally trained with an expectation-maximization algorithm which needs as hyperparameters the number of components. This limitation is discussed in section 4.3 which reviews the GMMs training algorithm with an unknown number of components.

Saliency Map

The human visual system is very efficient in processing complex scenes. One of the features that make our visual perception efficient, is the ability to focus on salient elements in a visual scene. Saliency qualify for human, what attracts his attention or gaze. Generally, interesting objects are those with salient features.

In computer vision, a family of methods, called *salient object detection*, aims at automatically determining the most salient element in a picture. The outcome of such methods is a saliency map in which pixels are labeled as salient or not. A saliency map is usually shown as a binary map in which a white area represents the most salient object in the picture. The works in this field are interested in the human visual system and thus, to emulate it thanks to machine learning and image processing techniques. So, assumptions relative to human perception are formulated.

A robot with this ability would be more efficient in the analysis of complex scenes. The relevance map methods presented in chapter 5 draws inspiration from salient object detection methods with the aim of removing the assumptions specific to human perception. A review of this field is described in section 5.3.

Affordance Learning in Robotics

Learning a visual representation with the help of interactions with the environment, as proposed by interactive perception, draws a direct link with affordances. An affordance, a concept introduced by Gibson [1966, 1979], is a relational property of the agent-environment system. In robotic, it is usually defined as a relationship between an object, an action, and an effect. This concept is practical to formalize a representation of the environment according to the robot internal structure and capabilities (Sahin et al. [2007], Zech et al. [2017]).

Works that aims to learn affordances in a robotic system are reviewed in section 7.2

2.4.2 Putting it all together

A few studies include all the domains introduced in the previous section. The works of Ŭgur et al. [2007] and Kim and Sukhatme [2015] address the issue of learning online affordances with robotic exploration.

Ŭgur et al. [2007] proposed a method for learning "traversability" affordance with a wheeled mobile robot which explores a simulated environment. The robot tries to go through different obstacles: laying down cylinders, upright cylinders, rectangular boxes, and spheres. The laying down cylinders and spheres are traversable while boxes and upright cylinders are not. The robot is equipped with a 3D sensor and collects data after each action labeled with the success of going through the objects. The sample data are extracted thanks to a simulated RGB-D camera. Then, an online SVM (Bordes et al. [2005]) is trained based on the collected data. The resulting model predicts the "traversability" of objects based on local features. To drive the exploration, an uncertainty measure is computed based on the soft margin of the model decision hyperplane. Finally, they tested their method on a navigation problem, on real robots and in a realistic environment. They demonstrate, by using the model learned in simulation, that the robot was able to navigate through a room full of boxes, spherical objects and cylindrical objects like trash bins without colliding with non-traversable objects.

Kim and Sukhatme [2015] in the same idea seeks to learn pushable objects in a simulated environment using a PR2 with an RGB-D camera. The objects are blocks of the size of the robots. They are either pushable in one or two directions, or not pushable. The PR2 uses its two arms to try to push the blocks. The learning process relies on a logistic regression classifier and a Markov random field is used to smooth spatially the predictions. The robot explores then the environment and collects data by trying to push the blocks. The outcome of the framework is what they called an affordance map indicating the probability of pushability of a block. When in Ŭgur et al. [2007] the learning is made on continuous space, in Kim and Sukhatme [2015] the environment is discretized in a grid with the cells of the size of a block, thus, the learning space is discrete. Finally, they use an exploration strategy based on uncertainty reduction to select the next block to interact with.

Both works are close to the work presented in this dissertation. They gather in a single study affordance learning, online learning, query strategies or exploration process, and interactive perception. They do not mention saliency map explicitly but the affordance map of Kim and Sukhatme [2015] is close from saliency map by the way they both segment interesting elements for the agent. Exploration and learning were conducted in simulation only and in simple environments. In this thesis, we present an approach based on similar principles but to learn directly in reality and on more complex and realistic environments. Moreover, their system was applied with only one affordance, in chapter 7, the proposed approach permits to learn a relevance map from several affordances.

2.5 Conclusion

The approach presented in this thesis is original in the way that it gathers the domains described in the previous section. There are only a few works who cover these different domains. Even if these works are conceptually similar to our approach, our goal is different as we have chosen to consider a real robotic setup. We have considered a two arms robot and real environments. The sum of the complexity of vision, of two 7 degrees-of-freedom arms with a minimum of context-specific assumptions creates an interesting challenge.

The main goal of the work proposed in this dissertation is to provide a modular framework to learn relevance maps from different affordances to bootstrap a developmental cognitive system.

Chapter 3

Background

Contents

3.1 Introduction

This chapter presents the theoretical background of Gaussian mixture models (GMM) and image processing methods used in this thesis. The collaborative mixture models (CMMs), the classifier introduced in chapter 4, is based on GMMs and on a geometrical analysis of multivariate normal distributions (MVND) (see section 3.2). The relevance map learning framework (see chapter 5) uses a method of supervoxels segmentation called voxel cloud connectivity segmentation and extracts visual features to feed the classifier based on color histograms in CIELab encoding and fast point feature histograms (FPFH). Both aspects are explained in section 3.3.

3.2 Gaussian Mixture Models

3.2.1 Classical GMM

GMM is a probabilistic regression method to estimate unknown probabilistic distributions from a dataset. A mixture model is a weighted sum of probabilistic distributions. Any distribution could be used, such as gamma or Poisson distributions, but Normal distributions are the most used in the literature. In multidimensional features space, Multivariate Normal Distribution (MVND) is used.

Multivariate Normal Distribution (MVND) is a probabilistic distribution with a multivariate Gaussian function as density function:

$$P(X|\mu, \Sigma) = \frac{exp(-\frac{1}{2} * (X - \mu)^T * \Sigma^{-1} * (X - \mu))}{\sqrt{|2\pi\Sigma|}} \tag{3.1}$$

Where X is a random variable, μ the mean and Σ the covariance of the MVND.

As we deal with real data, the covariance matrix is not always invertible. In this case the MVND is not defined. Therefore, a degenerate case of MVND can be used defined with the pseudo-inverse of the covariance and its pseudo-determinant:

$$P(X|\mu, \Sigma) = \frac{exp(-\frac{1}{2} * (X - \mu)^T * \Sigma^{+} * (X - \mu))}{\sqrt{det^*(2\pi\Sigma)}} \tag{3.2}$$

Where Σ^+ is the pseudo-inverse of covariance and det^* the pseudo-determinant. The pseudo-inverse of the covariance and the pseudo-determinant is computed thanks to the singular value decomposition (see appendix A).

In both cases, Σ^{-1} or Σ^+ has to be positive semi-definite, otherwise, the MVND is not defined.

The density function of an MVND is noted in the rest of this thesis $G(\mu, \Sigma, X)$.

A Gaussian Mixture Model (GMM) is a weighted sum of normal distributions, multivariate in our case.

$$\Gamma(W, \Theta, X) = \sum_{j=1}^{K} w_j * G(\mu_j, \Sigma_j, X) \tag{3.3}$$

3.2.2 Geometrical analysis of Multivariate Normal Distribution

Ellipsoid of tolerance In statistics, a way to study a dataset is to determine the confidence and tolerance regions of a distribution. While the confidence region can be used to estimate the parameters of the MVND, the tolerance region provides information on the whole dataset. For an MVND, the tolerance region is an ellipsoid. The size of the ellipsoid is determined by the level of confidence α which means all the data with a probability greater than $1 - \alpha$ is within the ellipsoid (for the confidence ellipsoid the probability must be greater than α). Generally, the level of confidence is low (around 5 %). The tolerance ellipsoid is a good approximation of the MVND and allows us to perform a geometrical analysis.

If the theoretical covariance matrix is considered, the distance between a point and the theoretical mean in the sense of Manahobolis follows a χ^2 distribution :

$$n(\mu^* - \mu)^T \Sigma^{-1} (\mu^* - \mu) \sim \chi^2_p \tag{3.4}$$

Indeed, χ^2 distribution is defined as the sum of squared Gaussian vectors : $\chi^2_p \sim \sum_i^p y_i^T y_i$. By the following variable change $Y_i = \Sigma^{-\frac{1}{2}}(\mu^* - \mu)$, the equation 3.4 is obtained.

But more interestingly, if Σ is unknown, we can use the sample estimation of the covariance and then rely on the Hotelling's T^2 distribution to approximate the distance between a point and the theoretical mean :

$$T_p^2(n) = nX^T M^{-1} X \tag{3.5}$$

Where X is a random variable following a normal distribution and M a matrix following a Wishart distribution. Moreover, the Hotelling's T^2 distribution and the Fisher distribution have the following property :

$$T_p^2(n) = \frac{np}{n-p+1} F(p, n-p+1) \tag{3.6}$$

From the equation 3.5 and 3.6, it can be deduced the following proposition :

$$\begin{aligned} T_p^2(n-1) &= \frac{(n-1)p}{n-p} F(p, n-p); \\ T_p^2(n-1) &= (n-1)X^T M^{-1} X \\ \Rightarrow X^T M^{-1} X &= \frac{p}{n-p} F(p, n-p) \end{aligned} \tag{3.7}$$

Which leads to the equation of the ellipsoid of confidence of an MVND :

$$\frac{n-p}{p} (\mu^* - \mu)^T \Sigma^{*-1} (\mu^* - \mu) = F(p, n-p) \tag{3.8}$$

This result is true when μ^* is the sample mean and Σ^* the sample covariance matrix of a dataset because $(\mu^* - \mu)$ follows a normal distribution and $n\Sigma^*$ follows the Wishart distribution :

$$\begin{aligned} (\mu - \mu^*) &\sim N_p(\mu, \Sigma) \\ n\Sigma^* &\sim W(n-1, \Sigma) \end{aligned} \tag{3.9}$$

We can also define the ellipsoid of tolerance as follows :

$$(X - \mu^*)^T \Sigma^{*-1} (X - \mu^*) = \frac{(n-1)p}{n-p} \frac{n+1}{n} F(p, n-p) \tag{3.10}$$

where n is the number of samples used to estimate μ^* and Σ^*; p the dimension of the features space; F the function of Fisher distribution. This result is obtained because $(X - \mu^*)$ follows a normal distribution and as before $n\Sigma^*$ follows a Wishart distribution :

$$\begin{aligned} (X - \mu^*) &\sim N_p(0, (1 + \frac{1}{n})\Sigma) \\ n\Sigma^* &\sim W(n-1, \Sigma) \end{aligned} \tag{3.11}$$

By replacing the Fisher variable by its quantil function we can obtain the confidence region of the MVND :

$$(\mu^* - \mu)^T \Sigma^{*-1} (\mu^* - \mu) \leq \frac{p}{n-p} F_{1-\alpha}(p, n-p) \tag{3.12}$$

Similarly, we can define the tolerance region :

$$(X - \mu^*)^T \Sigma^{*-1} (X - \mu^*) \leq \frac{(n-1)p}{n-p} \frac{n+1}{n} F_{1-\alpha}(p, n-p) \qquad (3.13)$$

By using the region marked out by the ellipsoid of tolerance defined in equation 3.13, we can apply a geometrical analysis. For instance, by testing if two ellipsoids are intersecting, we can determine if two MVND are overlapping each other.

Intersection of two ellipsoids Finding the intersection of two ellipsoids is a hard problem. To do so, the following equations system must be solved :

$$\begin{cases} k_1 & = & (X - c_1)^T A_1 (X - c_1) \\ k_2 & = & (X - c_2)^T A_2 (X - c_2) \end{cases} \qquad (3.14)$$

Where A_1 and A_2 are the axes of the two ellipses E_1 and E_2 with their center c_1 and c_2 and their constant k_1 and k_2.

Solving these equations would be too costly in terms of computation. The problem can be simplified by considering that the query ellipse E_1 intersects another ellipse E_2 if the center of E_1 is in the area defined by E_2. This simplification ignores some intersection cases but reduce significantly the computational cost. This leads to the equation 3.15 :

$$(c_1 - c_2)^T A_2 (c_1 - c_2) <= k_2 \qquad (3.15)$$

Transposed with the ellipses of tolerance of MVND, the equation 3.16 is obtained :

$$(\mu_1 - \mu_2)^T \Sigma_2 (\mu_1 - \mu_2) <= \frac{(n-1)p}{n-p} \frac{n+1}{n} F_{1-\alpha}(p, n-p) \qquad (3.16)$$

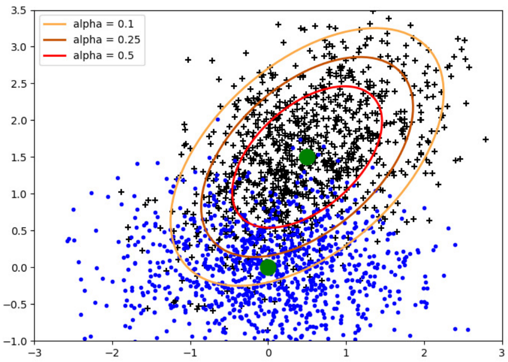


Figure 3.1: Two set of points drawn from two bivariate normal distributions. And for each three ellipses of tolerance for α equal to 0.5, 0.25 and 0.1. In this case, only with $\alpha = 0.1$ the intersection is considered between these two distributions.

Ignoring intersection cases is not harmful as the size of the ellipses of tolerance is controlled by the parameter α. As shown in figure 3.1, the smaller α, the bigger the ellipsoids is and the more intersections will be considered between two MVND. Equation 3.16 is a key condition in the training algorithm of the classifier (CMMs) proposed in chapter 4.

3.3 Image Processing

3.3.1 Supervoxels Segmentation

The work proposed in this thesis relies on an over-segmentation of a pointcloud. Over-segmentation is a segmentation in which each segment is smaller than the semantic parts of an image like objects, background, people, etc ... The relevance map, presented in chapter 5, is based on an over-segmentation into supervoxels using voxel cloud connectivity segmentation (VCCS) ([Papon et al., 2013a]). As illustrated in Figure 3.2, supervoxels are clusters of voxels. Voxels are the smallest unit of a 3D image as pixels are for 2D images. In our case, a voxel is a point as we work on pointclouds.

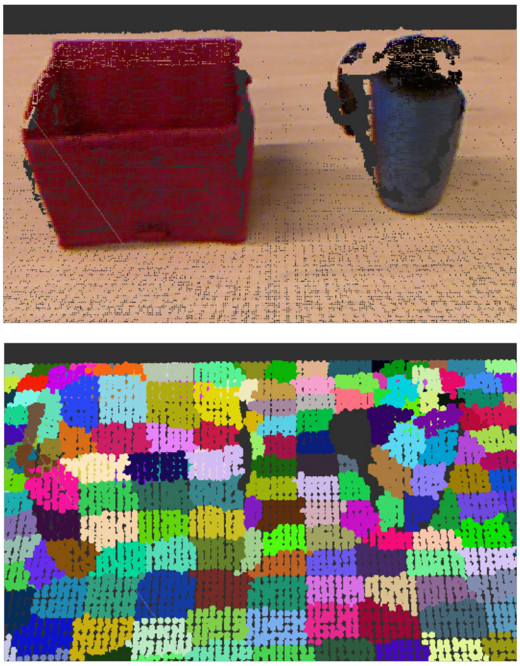


Figure 3.2: Examples of supervoxels segmentation. These images represent pointclouds, the top one is a pointcloud extracted from a kinect, the bottom one pointclouds represents supervoxels extracted on the top pointcloud. Voxels in these pictures are points as their are representing pointclouds.

VCCS is similar to superpixel methods such as SLIC superpixels (Achanta et al. [2010]) and turbopixel (Levinshtein et al. [2009]). It builds clusters of voxels directly on 3D pointclouds. The use of depth information to build supervoxels is a significant enhancement compared with superpixels methods because this segmentation respects object boundaries. The samples stored to update the classification are thus more likely to be associated with a single object; thus, it removes a significant source of noise in the classification. Moreover, VCCS works on all kinds of environments because the algorithm uses low-level features such as color, normals, and geometric descriptors (fast point feature histograms (FPFH) Rusu et al. [2009]). Therefore, VCCS produces a meaningful over-segmentation of RGB-D images.

VCCS is based on a region growing algorithm. At the beginning, voxel seeds are evenly distributed on the pointcloud. Then, a local nearest neighbors algorithm is applied to each seed controlled by a distance which combines color, spatial and shape distances (see equation 3.17). The nearest neighbor search is limited to a radius around the seed. Color distance is computed in CIELab color space and the shape distance is computed with FPFH.

$$D = \sqrt{\frac{\lambda D_c^2}{m^2} + \frac{\mu D_s^2}{3R_{seed}^2} + \varepsilon D_f^2} \tag{3.17}$$

Where D_c is the distance on color CIELab space divided by a constant m, D_s is the spatial distance divided by R_{seed} which is the maximal distance considered for the clustering and D_f is the distance on FPFH. For each of these distances, hyperparameters λ, μ and ε control the importance of color, spatial distance, and geometric similarity. This equation shows the four important hyperparameters of VCCS which control the size (R_{seed}) and the shape (λ, μ, ε) of the supervoxels. These last hyperparameters do not seem to be a limitation of the variety of environment on which this method can be applied as they are not environment-specific. For the experiments presented in chapters 5, 6, and 7, the parameters are fixed to the values proposed by Papon et al. [2013a]. However, the size of supervoxels is more critical as objects must be at least larger than a supervoxel.

Another drawback of VCCS methods is when extracted from a video stream on a static scene, the segmentation will change at each frame. This inconsistency of the segmentation prevents to apply supervoxels tracking methods.

In this work, supervoxels are the smallest visual unit considered. All visual features are extracted from a supervoxel and also the robot interacts with the supervoxels which have at least the size of its end-effector. We use the implementation of VCCS available in the PointCloud Library (PCL) (Rusu and Cousins [2011]).

VCCS algorithm output, like implemented in PCL is a set of supervoxels, with a centroïd point for each supervoxels. The centroïd point is at the average position and have the average color and the average normal of the supervoxels points. The output includes also an adjacency map representing the graph of geographical proximity of the supervoxels. Thus, to access to the neighbors of a supervoxel, it is enough to use the adjacency map.

3.3.2 Visual Features and descriptors extraction

Visual feature extraction is a key part of the framework proposed in this thesis. A visual feature characterizes one supervoxel. Two different kinds of features are extracted on the 3D pointcloud. Color histograms and fast point feature histograms

(FPFH) (Rusu et al. [2009]) which characterize the local geometry of a supervoxel. While color histograms are extracted from the RGB image, the FPFH are extracted from depth information provided by 3D cameras like stereoscopic cameras or active 3D cameras like microsoft kinect or asus xtion.

Before explaining color histogram features, *local invariant features* must be mentioned because of their popularity. Local invariant features are descriptors localized on keypoints. Their are extracted on the entire picture and produces keypoints which characterize salient parts. The most used feature is scale-invariant feature transform, better known as SIFT descriptor (Lowe [1999]), but there is also SURF (Speeded-up Robust Features : a lightweight enhancement of SIFT Bay et al. [2006]), BRIEF (binary robust independent elementary feature Calonder et al. [2010]), GFT (good feature to track Shi and Tomasi [1993]) and many others (Tuytelaars et al. [2008]). This family of descriptors is *local* because they create keypoints which characterize a small part of an image, and they are *invariant* because of their invariability on zoom, rotation, and translation. Thus, local invariant keypoints are very useful for images alignment, to make panoramic pictures for instance, or for object recognition and tracking.

But for our work, local invariant features cannot be used as output keypoints are localized and cannot be extracted anywhere in images. For instance, there would be just a few or no SIFT keypoints on a nontextured object. Because SIFT is based on contrast and difference on the images. A fully monochromatic image would have no SIFT keypoints at all.

Color Histogram Features

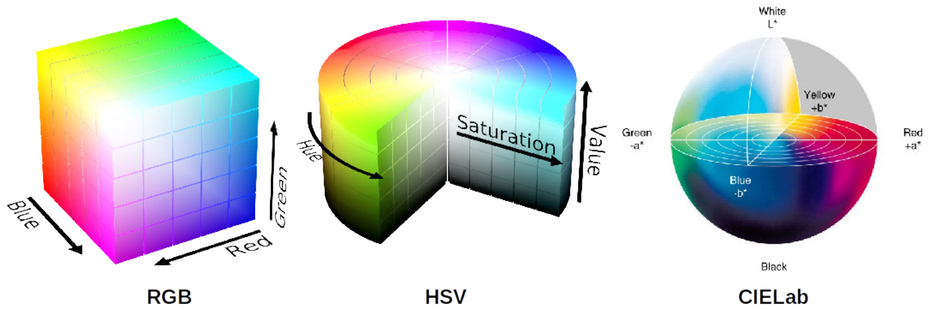


Figure 3.3: Representation of color spaces : RGB, HSV and CIELab

We are then more interested in histogram descriptors and particularly color histograms. Histograms are interesting because they downscale the quantity of information by controlling the resolution. First, the color space in which extracting the histogram needs to be chosen. In computer vision, most used color spaces are Red-Green-Blue (RGB), Hue-Saturation-Value (HSV) and Lightness-a-b (Lab). RGB is the default encoding of pictures in computer science but is not practical for computer vision. This encoding is practical for digital encoding, but it is not characteristic of any natural visual system and has some major drawbacks. For instance, a simple change of lightness would correspond to a change in the three dimensions of the encoding (see left part of figure 3.3), thus colors which seem close, are far in RGB encoding. HSV is better as it splits into three orthogonal dimensions the hue, saturation and value or lightness (for HSL) which is closer to natural vision. But HSV still has an artificial space structure (see middle part of figure 3.3). Finally, CIELab[1] is the most robust and realistic color encoding. Lab family encoding has been designed to be close to human vision. As it is shown on the right part of figure 3.3, Lab is structured on three dimensions, L for the lightness, a and b for the colors. CIELab encodes more colors than HSV and RGB.

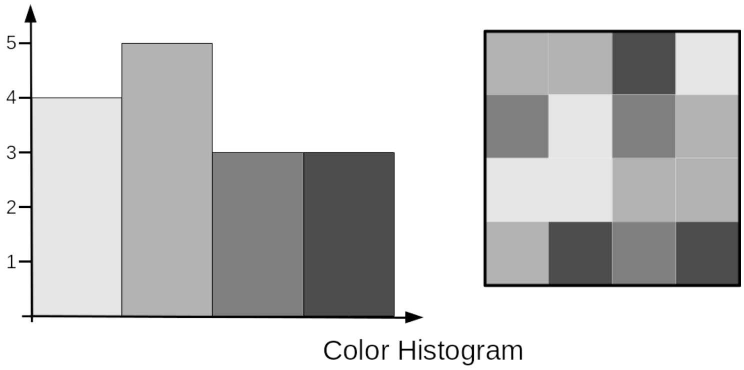


Figure 3.4: Illustration of a color histogram on one channel. The left part of the figure represents a histogram computed from the grayscale grid on the right part of the figure. The number of each gray level is counted to obtain the histogram.

A histogram consists in a division of space into intervals called *bins* and then counting the number of pixels belonging to each bin (see 3.4). Unlike local invariant features, histograms are global descriptors, it can be extracted on the whole image. It can also be local if it is extracted on a subpart of the image.

Fast Point Feature Histogram (FPFH)

FPFH is one of the most frequently used geometric descriptor (Rusu et al. [2009]). Among others, it is used in the computation of VCCS for oversegmentation in supervoxels (see section 3.3.1). It is generally used for 3D registration thanks to its high capacity for shape discrimination. Registration is a computer vision technique which consists in extracting discriminative features for recognition or images alignment. Figure 3.5 represents FPFH of different shapes.

FPFH is a simplification of PFH (point feature histogram) aimed at being faster to compute. FPFH is a combination of simplified PFH (SPFH) which are his-

[1] CIELab is an international standard of colorimetry decided during the International Commission on Illumination (CIE) of 1978

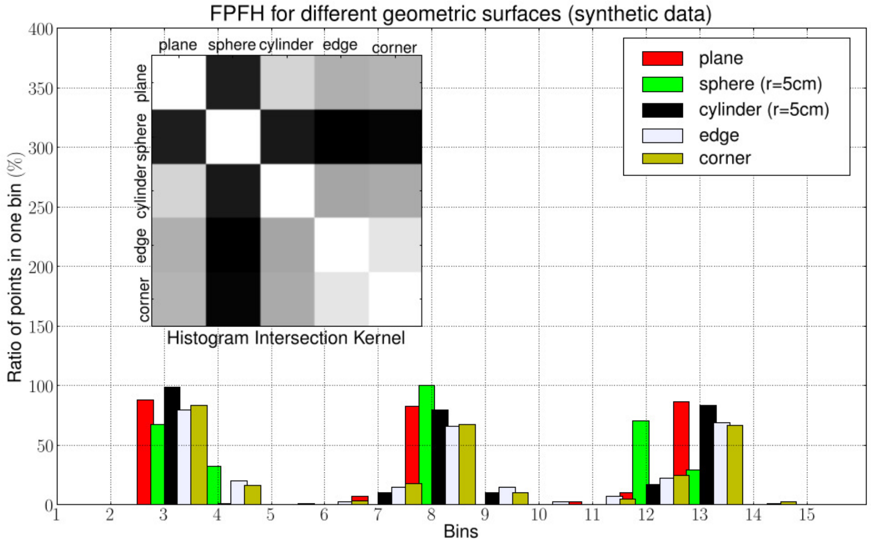


Figure 3.5: Example of FPFH for points lying on primitive 3D geometric surfaces. Figure taken from Rusu et al. [2009].

tograms of triplet of value (α, ϕ, θ) computed with equations 3.18.

$$\begin{aligned} \alpha &= v * n_t \\ \phi &= u * \frac{p_t - p_s}{\|p_t - p_s\|} \\ \theta &= arctan(w * n_t, u * n_t) \end{aligned} \tag{3.18}$$

Where $(.*.)$ is the scalar product, (u, v, w) is an orthogonal frame defined in equation 3.19 and represented in figure 3.6, n_t and n_s the normals to the surface at points p_t and p_s.

To apply these equations, a coordinate frame is defined at one of the points like shown in figure 3.6 and written in equation 3.19.

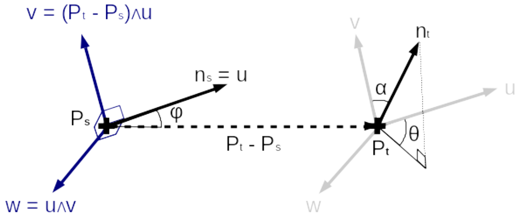


Figure 3.6: Schema of how the orthogonal frame (u, v, w) is defined on which the computation of SPFH is based. Figure reproduced from Rusu et al. [2009].

$$\begin{aligned} u &= n_s \\ v &= u \wedge \frac{p_t - p_s}{\|p_t - p_s\|} \\ w &= u \wedge v \end{aligned} \tag{3.19}$$

Where $(. \wedge .)$ is the vectorial product.

An SPFH is computed for the query point p_q and for its neighbors. The neighborhood of the query point is composed of the points into a sphere of radius r represented by the dashed circle in the 3.7. Thus, the computation of FPFH involves the neighbors of p_q and also the neighbors of the neighbors of p_q like shown in figure 3.7.

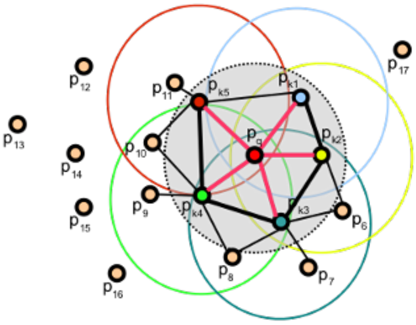


Figure 3.7: Schema of neighbors system for computing FPFH. Figure taken from Rusu et al. [2009].

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{w_i} SPFH(p_i) \tag{3.20}$$

Finally, FPFH is computed by summing SPFH according to the equation 3.20. Each SPFH are weighted with the distance w_k between the query point and its neighbors.

3.4 Conclusion

The methods presented in this chapter are used in different parts of the proposed framework. GMM is the basis of CMMs described in the next chapter, and the intersection condition of two MVND is a core component of this classifier. Supervoxel segmentation is used as downsampling method for feature extraction. Also, the targets chosen by the robot to interact with are supervoxels centroïds. This segmentation is the basis of the relevance map. Finally, the color histograms and FPFH are extracted to feed the classifier and build a relevance map. This is described in more details in chapter 5.

Chapter 4

Collaborative Mixture Models

The results and text of this chapter have been partially published in the following article :

> Le Goff, L. K., Mukhtar, G., Coninx, A., and Doncieux, S. (2019). Bootstrapping Robotic Ecological Perception from a Limited Set of Hypotheses Through Interactive Perception. arXiv preprint arXiv:1901.10968.

Other contributors:

- Stéphane Doncieux, Sorbonne Université (Thesis supervisor)
- Ghanim Mukhtar, formerly, in 2017, Sorbonne Université (Engineer)
- Alexandre Coninx, Sorbonne Université (Maitre de conférence)

Contents

4.1 Introduction

The main goal of this work is to develop a method to classify data collected during a robotics exploration labeled according to the success of an action primitive. This classification method must be able to handle a large variety of environments. Thus, the classifier has to generalize and adapt to different environments. The values of its hyperparameters should not be specific to a particular environment. In a complex environment, the classes may not be convex and the training dataset extracted from it may not be linearly separable. Furthermore, the classifier has to provide a measurement of uncertainty, which will be used for the exploration process. The output of the classifier needs then to be a probability rather than a single net value. The choice of the machine learning algorithm has then to fulfill the following criteria :

1 Handle non-convex/non-linearly separable dataset and feature space

2 Uncertainty measurement for query strategy

3 Have hyperparameters that are not specific to a particular environment

4 Supervised and adapted to classification

5 Online training

To classify the collected samples, a new classification method is introduced: the Collaborative Mixture Models (CMMs). This classification is supervised as the system labels the gathered samples thanks to the interactions of the robot with the environment and the effect detector. The training algorithm follows the *active learning* framework (see sections 2.3.2 and 4.4): samples are collected and labeled by the robot interaction, then added to the dataset. The next sample is chosen thanks to a query strategy based on *uncertainty estimation.* The collected samples are represented as a set of clusters for each class encoded by a multivariate normal distribution (MVND). They are summed to form a mixture model. There is, thus, a Gaussian mixture model (GMM) for each class. The parameters of the MVNDs are statistically estimated by using samples mean and samples covariance estimators. The number of components in each model is not given a priori and is adapted to the training set. Both mixtures start with one component and add or remove new components with *merge* and *split* operations. These operations adapt the number of components to the data. As one GMM is used for each class a geometrical analysis can be done to choose to merge two components or split one component (see section 3.2.2). MVND are approximated by hyper-ellipsoids of tolerance and the intersection between them is checked (see section 4.6.1).

We choose to use GMM as a basis for the classifier because GMM offers a suitable theoretical framework that meets the first and second criteria. GMM is a regression method known to be able to approximate a large set of probabilistic distributions (criterion 1). By encoding each class with a mixture, we have a classifier trained with a supervised learning algorithm that gives as output a probability of membership to the class (criteria 2 and 4). Also, a GMM can be seen as an ensemble learning method in which weak classifiers are combined to form a strong classifier. It is thus a classifier able to handle non-convex classes or non linearly separable dataset (criterion 1). Latent parameters of GMM are generally estimated by using an expectation-maximization algorithm which needs the number of components as a hyperparameter. This parameter depends on the environment as the more complex the scene in features space, the more components are needed. This is avoided in the proposed training algorithm by an online estimation of the number of components by split and merge operations. The number of components is thus no longer a hyperparameter of the classifier (criterion 3). Finally, by following active learning framework, samples are processed one-by-one; which corresponds to online learning (criterion 5).

The rest of this chapter is structured as follows : sections 4.2, 4.3 and 4.4 present related works about online learning, GMM algorithms with an unknown number of components, and query strategies in active learning which correspond to the three main features of CMMs; then section 4.5 defines formally the classifier and section 4.6 explains in details the training algorithm. The classifier is first evaluated in the next chapter on a robotic platform, and then an extensive study of the influence of the different components of the classifier is presented in chapter 6.

4.2 Online Learning

Ref	Type	non-convex	non-linear	uncertainty	environment specific hyperparameters	supervised
Cauwenberghs and Poggio [2001]	SVM	yes	yes	no	kernel, soft margin	yes
Bordes et al. [2005]	SVM	yes	yes	yes	kernel, soft margin	yes
Bordes and Bottou [2005]	SVM	yes	yes	no	kernel, soft margin	yes
Tax and Laskov [2003]	SVM	yes	yes	no	kernel, soft margin	no
Oza [2005]	bagging, boosting	no	yes	no	no	yes
Saffari et al. [2009]	random forest	yes	yes	no	no	yes
Saffari et al. [2010]	boosting	no	yes	no	no	yes
Cappé and Moulines [2009]	mixture model	yes	yes	yes	number of components	yes
Kristan et al. [2008, 2011]	mixture model	yes	yes	yes	level of compression	yes
Our Approach	GMM	yes	yes	yes	intersection sensibility (α)	yes

Table 4.1: Online learning algorithms review.

A learning algorithm for a classification problem can be seen as two people playing a game. One has a hidden proposition which must be guessed by the second one. Let us call the first one the expert and the second one the guesser. The expert gives clues to the guesser about the proposition. From the clues, the guesser must build hypotheses about the proposition. Two variants can be played. First, the expert writes a certain amount of clues and gives them to the guesser. The guesser studies them alone as much time he wants and finally gives to the expert his hypotheses. Let us call this variant the *offline* game. In the second variant, the expert tells directly to the guesser one clue at a time and the guesser must give directly after a first hypothesis. Then the expert tells to the guesser another clue and the guesser gives a new hypothesis to the expert. They continue like this until a certain amount of iterations. Let us call this last variant the *online* game. Of course, the *online* game is much harder for the guesser than the *offline* game. Because in the *online* game the expert can change his proposition during the game, in the case of an adversarial expert, which is not possible in the *offline* game. However, the expert and the guesser can have extra interactions during the *online* game which can help the guesser. Finally, the expert and his proposition can be seen as a classification problem on a certain feature space and the guesser as a classifier (Shalev-Shwartz et al. [2012]). Of course, this image is not specific to classification problems and can for instance work for regression too. But in this thesis, we give a focus on classification.

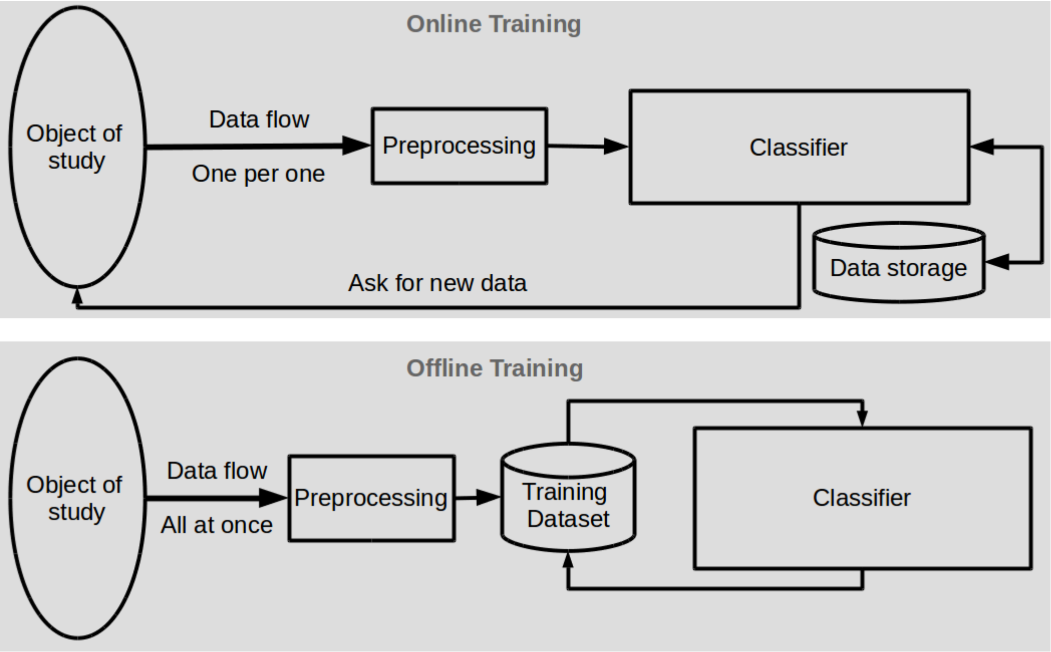


Figure 4.1: Dataflow of offline learning versus dataflow of online learning

A classifier trained online in a supervised way will process samples "on arrival", i.e. the data are processed through only once. One iteration corresponds to a "new" sample added in the dataset. At each iteration, the algorithm should be able to produce estimations or predictions. If the algorithm is made in an incremental fashion, there is no need for the data to be stored but this is not mandatory for online learning. In other words, online learning algorithms have to be able to answer a question based on previous information and newly just arrived information, the information is always partial and the question could change. Online learning is opposed to batch or offline learning in which all the data for training are available, the data can be processed several times, and the question is static. In much real life problems, online learning is more natural than batch learning because it is difficult to have entire control of the dataset or of the structure of the problem. On such problems having a complete dataset is difficult or impossible. The data flow of online and offline learning is illustrated in figure 4.1.

Online learning is useful for streaming data processing in which the order of arrival of the data could not be controlled, like in weather prediction or financial modeling. Or, when a sampling process is used to gather a meaningful dataset, like in autonomous robotics. In this case, the sampling process needs information from the classifier trained on the dataset.

According to the literature, the important features of online learning algorithms are :

- Estimation of the classification, prediction, or error must be done *on-the-fly* (mandatory)
- Incremental update of the classifier (optional)
- Convergence of the online training algorithm must drive to the same classifier as batch training (optional)
- Capacity of adaptation to the new incoming data (optional):
    - Estimation of the relevance or the importance of a sample
    - Unlearning process

Most of the classification algorithms are trained in batch in their classical form. This section will list some of the main classification methods converted into online learning. So, all the methods described in the following section meet the 5th criterion.

4.2.1 Support Vector Machines

Support Vector Machines (SVM) construct a hyperplane or set of hyperplanes to separate data from different classes in a features space. SVM can be formulated as an optimization problem in which the parameters of a quadratic function must be found to separate two sets of data. To be able to solve a non-linear problem a kernel is used in the quadratic function. So, SVM is a kernel classifier trained in a supervised way.

To train an SVM online, a samples removal step must be implemented to avoid overfitting. A measurement of relevance or importance of a sample is useful for sample removal. For instance, *leave-one-out* process is used in incremental SVM to remove samples (Cauwenberghs and Poggio [2001]). Most of the proposed online SVM algorithms converged toward the same solution as SVM trained offline batch such as LASVM or the Huller (Bordes and Bottou [2005], Bordes et al. [2005]). Online SVDD is an adaptation of SVM to unsupervised and online learning (Tax and Laskov [2003]).

By definition, SVM fulfill the first and the fourth criteria. The third criterion is more problematic with SVM because of the choice of the kernel type and the soft-margin parameter. The kernel, as well as the soft-margin parameter, must be chosen with respect to the problem (Burges [1998], Smits and Jordaan [2002]).

An uncertainty measurement (2nd criterion) could be obtained by using the soft-margin, which defines a soft decision hyperplane. The soft-margin could define an uncertainty on the classification area (Bordes et al. [2005]). Of course, the parameter of the soft-margin must be well chosen between precise classification and large exploration areas.

4.2.2 Bagging, Boosting and Random Forest

Boosting and random forest are part of the ensemble learning family of methods. These methods use a set of weak classifiers and combine them to produce a strong classifier, i.e. a classifier able to separate non-linear dataset and solve non-convex classes. Random forest combines randomly generated trees by training them on a subset of the data. Boosting can be used with different kinds of weak classifiers and combine them by weighting them and optimizing these weights. Bootstrap aggregating (bagging) algorithm is a method to generate, from a unique dataset of size N, M datasets of size N. The samples are distributed by sampling uniformly with replacement the training dataset which means that each dataset contains duplicate samples. Boosting such as AdaBoost or random forest relies on bagging.

In classical bagging, the data are divided following a binomial distribution. In online case, all data are not available beforehand and must be distributed on-the-fly. The Poisson distribution is a good choice as the binomial distribution tends toward Poisson distribution when the dataset size grows to infinity. So, model learning algorithms trained with batch learning or online bagging converge to the same classifiers as the training set grows to infinity (Oza [2005]).

Saffari et al. [2009, 2010] proposed an online random forest and an online boosting algorithm. These algorithms are used to track faces in videos in real time. In this task, exploration is not needed; thus, estimation of the uncertainty or confidence of the classification to drive a sampling process is not required. As such, this issue is not addressed. One option for measuring uncertainty is to use the confidence of the classification of a tree used in the random forest algorithm. This confidence is used to estimate which tree will give the best prediction for a given sample. Online random forest of Saffari et al. [2009] featured an unlearning process by pruning trees or discarding entire trees. They use the *out-of-bag* error to evaluate which tree must be pruned or discarded. This error is computed by evaluating the tree on samples which were not used for its training. These samples are called left out the bag, hence the name of the error: *out-of-bag*. For boosting, to the best of our knowledge, unlearning does not seem to be addressed. This is due to the fact that online boosting does not need such process to be efficient.

Craye et al. [2015] used the online random forest proposed by Saffari et al. [2009] as a classifier and intelligent and adaptive curiosity (IAC) (Oudeyer [2004]) for the exploration process. IAC does not involve uncertainty estimation, but it is based on maximizing learning progress; which requires to segment environment into areas large enough to compute the learning progress. Also, to compute meaningful learning progress in each area, assumptions about the environment must be stated. In the study of Craye et al. [2015], the segmentation is made by an expert, each area represents a room or a corridor in a building. Thus, the robot needs a lot of prior information about the environment to use this exploration process.

4.2.3 Mixture Models

Mixture models are probabilistic learning methods used for regression. A mixture model is generally trained offline with expectation-maximization (EM) algorithms. Thus, a new algorithm is needed to train mixture models online.

A first approach is to adapt EM to online learning. With this aim, for instance, E-step can be replaced by a stochastic approximation of expectation and let M-step unchanged (Cappé and Moulines [2009]). This online EM algorithm meets the first, second and fifth criteria. Indeed, mixture models can handle non-convex spaces and have an inherent uncertainty measurement as it is based on probabilistic distributions and naturally it is online. But this algorithm still needs the number of components as hyperparameter which does not meet the third criteria. Mixture models are regression methods (forth criteria) but can easily be converted into classification methods (see section 4.5).

Kristan et al. [2008, 2011] proposed an incremental parameter estimation for multivariate and univariate GMM by drawing inspiration into kernel density estimation (KDE) for regression. Density estimation methods aim at estimating a density function from a dataset in which the data are assumed to be drawn from an unknown density function. Several approaches are applied such as mixtures of histograms, mixture of naive estimators, or mixture of kernels. If a Gaussian distribution is used as kernel then the density will be estimated thanks to a GMM. The idea of Kristan et al. [2008, 2011] is simple: for each new incoming sample, a new component is added to the mixture centered on the new sample. At a fixed period a compression operation is applied. The compression consists of a clustering of the components by minimizing the local error of clustering and the number of clusters. The local error of clustering is the error of prediction of the new components after clustering.

But the compression step can be computationally heavy especially in the multivariate case. Also, the frequency of the compression step must be well chosen as a compromise between the precision the model in relation to the training data and the generalization ability of the model. If too few compression steps are applied the model will overfit the data because there will be too much components, but if too many compression is done, the model will be inaccurate on the training data.

In a similar idea, Declercq and Piater [2008] proposed a regression method based on a bivariate GMM. The model is trained online and the number of components is estimated thanks to split and merge operations. Like in Kristan et al. [2008, 2011], a new data is added to the model as a Gaussian component with a prior covariance representing the observation noise. This new component is merged to the most uncertain component in the existing mixture. The uncertainty of a component is estimated thanks to an uncertain Gaussian model which provides a quantitative estimate of the ability to describe data that follows a Gaussian distribution. This measure is called *fidelity*. If the fidelity of the newly merged component is below a certain threshold then this component is split into two new components. The parameters of the new components are estimated via expectation-maximization.

The proposed methods of Kristan et al. [2008, 2011], Declercq and Piater [2008] are interesting as they associate online learning with an estimation of the number of components, but they do not address the question of the sampling process. They also consider regression problems whereas in this thesis the problem is formulated as a classification problem. In a classification problem the output space is discrete, e.i., the labels, while in regression the output space is continuous. In our opinion, considering labels suit more to our problem which is, in the next chapter, discriminate features between those characterizing moveable elements and those characterizing non-moveable elements. When, for instance, regression is practical for estimating trajectories or continuous signals.

Finally, there is no existing method in the literature that fits all of our criteria. Random forest may be a good choice as it is efficient and general; however, it is not an algorithm designed to drive the exploration of an unknown feature space because of its lack of uncertainty measurement. Existing methods based on mixture models are satisfying as they have an inherent uncertainty measurement. Thus, mixture models could give an appropriate classifier to design an exploration process of an unknown environment thanks to the corresponding probabilistic framework which gives tools to measure uncertainty, but it needs to select beforehand the number of components which is problem specific. Nevertheless, previous studies have been conducted on mixture models with an unknown number of components. The next section reviews these studies.

4.3 Gaussian Mixture Models with an unknown number of components

In general, to estimate the parameters of a mixture model, the expectation-maximization algorithm (EM) is used. But in order to use the EM algorithm, the number of components, i.e. the number of summed distributions, must be provided. It is a hyperparameter specific to the complexity of the problem. This section reviews works which try to estimate the number of components.

Vlassis and Likas [2002] proposed a greedy version of the EM algorithm. They assume that training a mixture with an unknown number of components can be achieved by directly maximizing the loglikelihood. At each step, they add a new component by choosing a point in the dataset by maximizing a utility function (the loglikelihood summed with an approximation of the second order Taylor development of the loglikelihood). Then they use a partial EM algorithm to estimate the parameters of the new component and by keeping fixed the rest of the mixture. The main drawback of the method is the assumption of achieving a good density estimation by only maximizing the loglikelihood which is valid in offline learning. But with online learning, such a hypothesis does not hold anymore because the training dataset is only partially representative of the problem. Which means that the previously built GMM could not be good anymore, therefore a process of unlearning or editing the previously added components is needed. Loglikelihood is then not a sufficient optimization criterion.

Richardson and Green [1997] proposed an algorithm called reversible jump Markov chain Monte Carlo (RJMCMC) for univariate GMM with an unknown number of components which is based on the Metropolis-Hasting update. Metropolis-Hasting update consists in updating the parameters according to an acceptance probability. This probability is based on a ratio between the density before and after a *move*, i.e. a variation of a parameter and also the probability of making this *move*. The issue of GMM with an unknown number of components is that the space of parameters have variable dimensions and classical MCMC are made for fixed dimensionality. Among classical moves (updating weights, updating MVND parameters), RJMCMC has split/merge and birth/dead moves which consist in adding or removing components, and thus, extending or shrinking the number of parameters. So, these moves change the dimension of the parameter space.

Zhang et al. [2004] extend the study of Richardson and Green [1997] to multivariate GMM by using simplified GMM with a diagonal covariance matrix. The main difficulty of RJMCMC is preserving the property of the mean and covariance before and after the split/merge move which is easy for the merge (see equations 4.1) but not for the split operation. As the splitting of an MVND is an ill-posed problem, i.e. not well-posed in sense of Hadamard[1], a lot of free parameters need to be introduced to respect the properties of a covariance matrix (see equations 4.2), i.e. to be semi-positive definite.

$$\begin{aligned} w_i &= w_j + w_k \\ w_i * \mu_i &= w_j * \mu_j + w_k * \mu_k \\ w_i * (\Sigma_i + \mu_i * \mu_i^T) &= w_j * (\Sigma_j + \mu_j * \mu_j^T) + w_k * (\Sigma_k + \mu_k * \mu_k^T) \end{aligned} \tag{4.1}$$

Equations 4.1 correspond to the update of the MVND parameters after the merge of two components of indexes j and k into a component of index i (Zhang et al. [2003]). Σ and μ are the covariance matrix and mean of a component and w its weight in the mixture.

[1] A well-posed problem defined by J. Hadamard is a problem with an existing unique solution which changes continuously with the initial conditions

$$
\begin{aligned}
& w_j = w_i \alpha, \quad w_k = w_i(1-\alpha) \\
& \mu_j = \mu_i - \sqrt{\frac{w_k}{w_j}} u a_l, \quad \mu_k = \mu_i + \sqrt{\frac{w_j}{w_k}} u a_l \\
& \Sigma_j = \frac{w_k}{w_j}\Sigma_i + (\beta - \beta u^2 - 1)\frac{w_i}{w_j} a_l a_l^T + a_l a_l^T \\
& \Sigma_k = \frac{w_j}{w_k}\Sigma_i + (\beta u^2 - \beta - u^2)\frac{w_i}{w_k} a_l a_l^T + a_l a_l^T
\end{aligned}
\tag{4.2}
$$

Equations 4.2 presents the update of MVND parameters after the split of component i into two components j and k, as proposed by Zhang et al. [2003]. Equations 4.2 come from the singular values decomposition (see appendix A) of the covariance matrix. These equations keep the covariances symmetric positive definite and thus allow us to define the new MVND. Moreover, a_l is the l^{th} axis of the hyper-ellipsoid of tolerance of the MVND to be split. That allows the algorithm choosing the direction in which the distribution will be split. The main drawback of this approach is the 4 new parameters introduced: α, β, u, l. l can be chosen randomly or the principal axis can be chosen. The best choice would be to select the direction which enhances most the regression. This issue is not addressed in the literature. α is related to the *importance* of the components in the mixture, u to the position of the new components and β *and* u to the shape of the distribution. These equations (4.1, 4.2) are equivalent to the one used by Richardson and Green [1997] for univariate GMM.

Ueda et al. [2000] proposed a split and merge EM algorithm based on a linear heuristic for merge and split operations. After a partial EM step, split-merge candidates are chosen and finally, another EM step is applied to the candidates. Zhang et al. [2003] extend their work by using the above equations (4.1, 4.2).

Other approaches, already described in the previous section, are proposed by Kristan et al. [2008, 2011], Declercq and Piater [2008] based on kernel density estimation. Their approach is similar to split and merge approaches except for Kristan et al. [2008, 2011] where they use only merge during the compression phase.

Most studies formulate the problem of estimating the number of components with split and merge operations. The mixture can start with one component and during the training add components thanks to the split operation and keep a reasonable number of components thanks to the merge operation. Or, add a component for each sample and use merge operation to reduce the number of components. Whereas a merge operation is a well-posed problem, a split operation is an ill-posed problem which forces to use intensive computation. All the methods presented in this section are for data analysis, "soft" clustering or regression. The problem in this thesis is a classification problem, so the labels are useful information to simplify the problem.

Methods using KDE (proposed by Kristan et al. [2008, 2011]) are well adapted to online learning and estimate the number of components, but the compression step can be expensive in term of computation. In online learning, it is better to divide as much as possible the computation among the iterations, which is not the case in this framework. It is more practical to use split and merge operations which could be computationally lighter than a compression step. Declercq and Piater [2008] proposed a similar approach that is computationally lighter, but they have tested their method only on a bivariate case. Kristan et al. [2008, 2011], Declercq and Piater [2008] have not addressed the question of the sampling process.

The next section reviews machine learning methods which proposed sampling processes called *query strategies*. This family of methods is called *active learning*.

4.4 Query Strategies in Active Learning

The use of query strategies aims at limiting the number of samples to label. In supervised learning, all the training data must be labeled because they are processed generally randomly, like for instance in batch learning. In active learning, the samples are labeled when they are queried and are assumed to be processed in an optimized, well chosen order. The classifier is then expected to converge with fewer training data with active learning than with classical supervised learning. Active learning data flow is illustrated in figure 4.2. In this section, different query strategies are reviewed with a focus on *uncertainty sampling*.

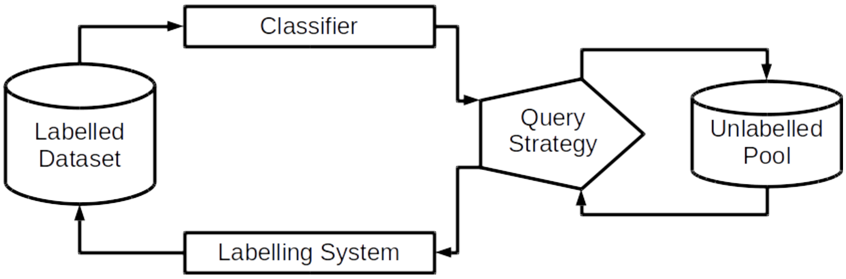


Figure 4.2: Active learning data flow: the classifier is trained on a labeled dataset; then a sample is queried in a pool of unlabeled data; a labeling system labels the chosen sample; finally, the newly labeled sample is added to the training dataset. Schema based on the study of Settles [2012]

4.4.1 Uncertainty Sampling

An easy and widely used family of query strategies is *uncertainty sampling* (Settles [2012]). Intuitively, uncertainty sampling allows the learning algorithm to choose samples where the classification is inaccurate. With a probabilistic classifier for a two classes problem, the strategy is simply to choose the samples that have a prediction close from 0,5 (Lewis and Catlett [1994]). With more than two classes, this simple approach is not enough.

For multi-classes problem, a criterion to choose which class to explore is necessary. A naive approach could be to select the class in which the prediction is the *least confident*. In the study of Culotta and McCallum [2005], they query the sample with the lowest prediction in its most probable label (see equation 4.3). The drawback of this query strategy is to consider only the most probable classes and ignore incorrectly encoded classes.

$$\begin{aligned} x^* &= argmax_x(1 - P_\theta(\hat{y}|x)) \\ \hat{y} &= argmax_y(P_\theta(y|x)) \end{aligned} \tag{4.3}$$

Where $P_\theta(y|x)$ is the probability of x belonging to class y.

To overcome this difficulty, Scheffer et al. [2001] proposed a margin sampling approach with a hidden Markov model. The idea is to consider the second most probable classes too. The algorithm will query samples with the smallest margin between its predictions of the two most probable labels.

$$x^* = argmin_x(P_\theta(\hat{y_1}|x) - P_\theta(\hat{y_2}|x)) \tag{4.4}$$

Equation 4.4 presents this query strategy where $\hat{y_1}$ and $\hat{y_2}$ are the two most probable labels for the sample x. The idea behind margin sampling is that if two classes are well separated, it means that the classification is probably accurate for these two classes. The opposite means an uncertainty which needs further investigation.

Another uncertainty sampling approach uses Shannon entropy (see equation 4.5) to query samples about which the classifier has few information. In the study of Holub et al. [2008], they use *minimum expected entropy* as a query strategy to train a classifier for object recognition. Similarly, Otte et al. [2014] use entropy to let a robot explore an environment and learn the degree-of-freedom of the different elements in the environment, like a door or a drawer. Entropy-based query strategies are general and theoretically satisfying but are often costly to compute. Interestingly, least confident, margin and entropy-based sampling methods are all equivalent to querying samples with probability to be part of a class close to 0.5.

$$H_\theta = \left(- \sum_i P_\theta(y_i|x) log(P_\theta(y_i|x)) \right) \tag{4.5}$$

Huang and Zhou [2013] proposed to combine uncertainty and diversity as query strategy. They use *label cardinality inconsistency* (LCI) (Li and Guo [2013]) which measures the difference between the number of predicted positive labels and the actual number of positive labels in the already labeled dataset. LCI is combined with margin sampling for multi-label classification. The main difficulty with LCI is as the number of labeled samples grows the difference tends to be very small. Thus, Huang and Zhou [2013] proposed to combine it with a diversity measure which favors querying samples in a label with the smallest dataset.

A parallel could be made with *intrinsic motivations* used in developmental robotics or psychology. Intrinsic motivations are often implemented as a reward function in the reinforcement learning framework. During an intrinsically motivated exploration, the agent produces events by interacting with its surrounding with the aim of building a world model. The choice of the next action to apply is based on a predictor which links an event as the consequence of an action and on a reward function which takes an event as argument. If the labeling system in the active learning framework is a robot collecting data by interacting with an environment, then the new sample collected is equivalent to an event. Thus, the reward function is equivalent to the query function. For instance, the artificial curiosity proposed by Oudeyer [2004] is an intrinsic motivation based on learning progress: the agent seeks for novel experiences for which its potential learning progress is not too high or too low. Therefore, this motivation drives the agent to always learn new things and to not stuck itself in too hard situations.

Another family of reward functions is based on uncertainty estimation (Oudeyer and Kaplan [2009]). A naive approach consists in pushing the robot to explore events with the lowest probability to occur, i.e. events it has rarely observed. But of course, such a motivation could quickly lead the robot into a dead end, where the robot tries to observe a very unlikely event. Another motivation is based on entropy, to push the robot to decrease the uncertainty about the environment by decreasing the entropy of an event, i.e. by increasing knowledge about this event. Finally, another motivation, called *surprise*, consists in exploring events which occur but are strongly expected to not occur. Such events increase the uncertainty about the environment and need to be explored.

As query strategies in active learning, reward functions are optimization value functions which guide the learning system. Active learning is then linked to interactive perception presented in section 5.2.

Uncertainty sampling is an intuitive exploration and query strategy and is suitable to use with probabilistic machine learning algorithms like GMM. Collaborative mixture models (CMMs) are based on GMM and the goal in the proposed thesis is to let a robot explores an environment to increase its knowledge about it, in other words the goal is to decrease uncertainty. Uncertainty reduction seems to be the natural query strategy for the proposed method. However other query strategies were proposed in the literature and are worth attention.

4.4.2 Other Query Strategies

Query-by-committee is a query strategy inspired by ensemble learning in which a certain number of models are trained to generate different hypotheses (Seung et al. [1992]). Other methods derived from boosting and bagging have been proposed by Mamitsuka et al. [1998]. They are calledquery-by-bagging and query-by-boosting. To apply such methods the models must be light to train and a disagreement measure between the classifiers is required. The generated hypotheses must be different enough for the query strategy to be efficient.

Other approaches consider how the model will change after processing new samples. *Expected gradient length* (EGL) is a query strategy based on gradient optimization (Settles [2012]) which uses the *gradient vector length* in the features space to find which sample will modify the model the most. To minimize the generalization error the model must change, so it makes sense to select samples which change the most the model although this change could decrease the classifier quality (Cai et al. [2013]). Moreover, samples which change the most the model, are the most informative about the problem, according to the current state of the model (You et al. [2014]). EGL is well adapted to gradient-based machine learning algorithm. But it is less practical with a GMM because estimating its gradient is complex.

Another approach consists in minimizing the number of incorrect predictions. This is called *expected error reduction* (EER) (Roy and McCallum [2001]). All the candidate models after adding each sample of the pool of unlabelled data and for each potential label are considered to find the model which minimizes the most the generalization error. This family of approaches can be very costly as a prediction must be estimated for each sample in the unlabelled pool times each potential label. Thus, EER will be useful for models with light estimations processes like Gaussian random fields and problems with few labels (Settles [2012], Zhu et al. [2003]).

Variance reduction query strategy consists in minimizing the output variance of the model. Variance reduction is linked to the expected error of the classifier. Thus minimizing the variance equates indirectly to minimizing the expected error (Geman et al. [1992]).

From another point of view, some want to make the model as close as possible from the data. This can be useful for regression, for instance. *Density-Weighted Methods* is a query strategy which aims at reducing the gap between the data and the model by selecting samples that are representative and avoid outliers.

Query strategies are defined in relation to the model chosen to represent the data. For instance, if a neural net is used in an active learning framework, it makes sense to use EGL as query strategy. As the classifier (CMMs) proposed in this chapter is probabilistic, it is straightforward to choose a query strategy based on uncertainty estimation. Moreover, uncertainty estimation does not need further computation as it is provided directly by the model, unlike EGL, EER, variance reduction or query-by-committee.

4.5 Definition of the classifier

CMMs is formalized with conditional probabilities and GMMs as base density distributions. In most of the experiments conducted for this thesis the classifier is used with two classes but in the following section, the classifier is defined in the general case of multi-class.

The classifier has the following parameters :

- N : number of classes
- K_l : current number of components of the mixture model of class $l \in [\![1, N]\!]$.
- $S = \{s_i, l_i\}_{i<I}$: dataset of samples s_i and with their corresponding label l_i.

- $\Theta_l = \{\mu_k, \Sigma_k\}_{k<K_l}$: MVND parameters of the mixture of class $l \in [\![1, N]\!]$ with μ_k the mean and Σ_k the covariance matrix.
- $W_l = \{w_k\}_{k<K_l}$: weights of the mixture model of class $l \in [\![1, N]\!]$.
- $L \in [\![1, N]\!]$: label asked to the classifier to be predicted.

Class Definition A class is a subspace of the feature space pointed out by a label. Equation 4.6 gives the probability for a sample X to be part of class i.

$$P(L = i|W, \Theta, X) = \frac{1 + \Gamma(W_i, \Theta_i, X)}{N + \sum_{l=1}^{N} \Gamma(W_l, \Theta_l, X)} \tag{4.6}$$

where $\Gamma(W_l, \Theta_l, .)$ is the GMM with parameters W_l and Θ_l; $W = \cup_{l=1}^{N} W_l$ and $\Theta = \cup_{l=0}^{N} \Theta_l$. To have a default probability of $\frac{1}{N}$ when all the GMMs are empty, one is added to the numerator and N is added to the denominator.

Equation 4.6 generates the output of the classifier.

Component Definition A component is a set of points of the feature space statistically represented by a multivariate normal distribution (MVND). A component is part of a class, i.e. all points part of a component have the same label or are a member of the same class. Equation 4.7 gives the probability for a sample X to be part of a given component i.

$$P(k = i|X, \Theta, l) = \frac{w_i * G(\mu_i, \Sigma_i, X)}{\sum_{k=1}^{K_l} w_k * G(\mu_k, \Sigma_k, X)} \tag{4.7}$$

Let us note $C_k(X) = (w_k, G(\mu_k, \Sigma_k, X)), S_k, l)$ a component and $M_l = \{C_k\}_{k<K_l}$ the set of K_l components of the class l. Where S_k is the set of samples used to estimate μ_k, Σ_k and w_k.

The weights of the mixture models are computed thanks to the equation 4.8.

$$w_k = \frac{|C_k|}{\sum_{i}^{K_l} |C_i|} \tag{4.8}$$

4.6 Algorithm

The Collaborative Mixture Models (CMMs) relies on a supervised learning algorithm. The algorithm builds one mixture per class. Each mixture is made up with several Gaussian distributions associated with a weight. Each distribution supports a cluster of samples, called a component (see the previous section).

CMMs could be trained in online mode or batch mode. The online mode consists in iterations in which a single new sample is added at a time, along with its label. Adding a sample consists of three main steps (see Algorithm 1 and figure 4.3):

1 If there is no component yet in the class of the new sample, create a new one, otherwise, find the closest component and add the sample to this component. Finally, update the parameters of the component;

Algorithm 1 update CMMs algorithm

```
procedure UPDATE CMMs($(s, lbl), M_1, ..., M_N$)
    for iter = $1 \to T$ do
        Add the new sample $(s, lbl)$ to the model :
        if $M_{lbl} = \emptyset$ then
            $C \leftarrow \{w, G(s, cI, .), \{s\}\}$  ▷ Create a new component with $I$ identity matrix and $c$ a constant
            $M_{lbl} \leftarrow C$
        else
            $C \leftarrow closest\_component(s, lbl, M_{lbl})$  ▷ Find the closest component from $s$ with label $lbl$
            Update the parameters of $C$ with the new sample $s$
            if SPLIT($C$,l,$M_1, ..., M_N$) is not successful then
                MERGE($C$,l,$M_1, ..., M_N$)
        for $l \in [\![1, N]\!]$ do
            $C \leftarrow random\_choice(M_l)$ ▷ Randomly choose a component from $M_l$
            if SPLIT($C$,l,$M_1, ..., M_N$) is not successful then
                MERGE($C$,l,$M_1, ..., M_N$)
    return $M = \cup_{l=1}^{N} M_l$
```

2 a *split* operation is applied to the updated component. If it is not successful the *merge* operation is then applied;

3 one component per class is randomly chosen and the *split* operation is applied to each of them. If the selected component is not split then the *merge* operation is applied.

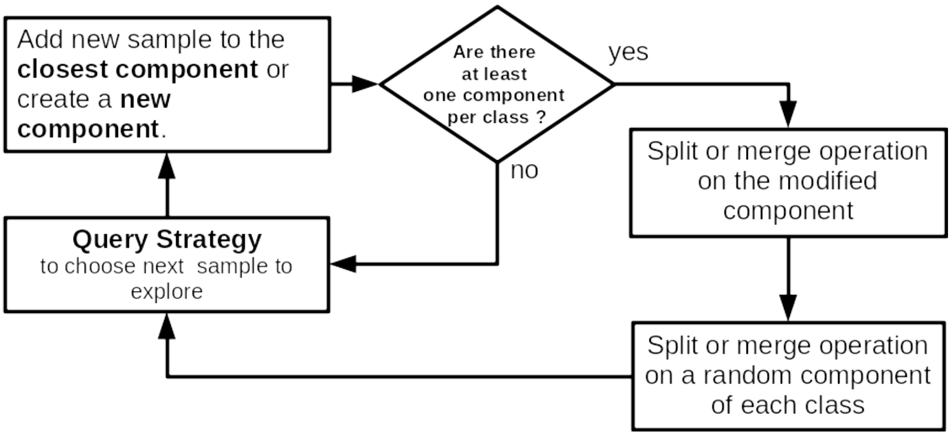

Add new sample to the **closest component** or create a **new component**.
Are there at least one component per class ?
yes
no
Split or merge operation on the modified component
Split or merge operation on a random component of each class
Query Strategy to choose next sample to explore

Figure 4.3: Online training algorithm schema of CMMs

Batch mode is structured into epochs in which a batch from the training dataset is processed. An epoch consists of two steps :

1 Adding all samples from the batch to the GMM of its label. As in online mode, each sample is added to the closest component.

2 Split and merge operations are applied to each component of the CMMs. As in online mode, the merge operation is applied only if the split operation has failed.

4.6.1 Split and Merge operation

The goal of the *split* operations is to keep only convex components. Indeed, MVNDs are only relevant to model convex spaces or set of data, as they can be represented as hyper-ellipses. *Merge* operation aims at reducing the number of components to avoid overfitting and reduce computational cost. Figure 4.4 illustrates the cases where *split* and *merge* operations are applied :

- **Split Case :** When two components of different classes are crossing or intersecting, one of these components is non-convex. This component must be split into two new components.
- **Merge Case :** When two components of the same class are crossing or intersecting, they can be merged to form a bigger convex component.

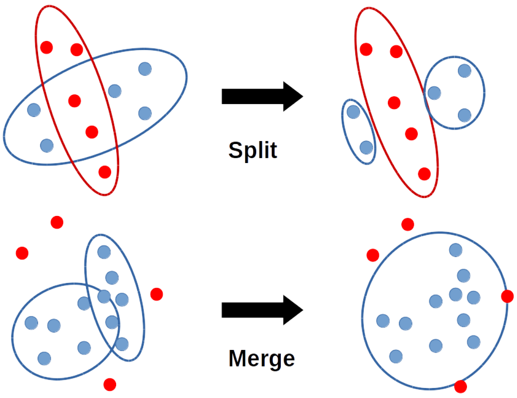


Figure 4.4: Schema of cases where the split and merge operations are applied. The top part, if two components of different classes intersect, one of the components should be split. The bottom part, if two components of same classes intersect, they should be merged.

Components intersection The *split* and *merge* operations are based on a geometrical interpretation of the components. An MVND can be represented geometrically as an ellipsoid. The proposed geometrical interpretation relies on the ellipsoid of tolerance of the distribution as explained in section 3.2.2. An ellipse of tolerance represents the area in which all the points have a probability greater than $1 - \alpha$ to be in the corresponding component. The axes of the ellipsoid are the eigenvectors of the inverse of the covariance matrix. In both cases (merge and split), there are two components that intersect each other. The parameters of the ellipsoid of tolerance can thus be computed as follows:

$$(X-\mu)^T\Sigma^{-1}(X-\mu) = \frac{(n-1)p}{n-p}\frac{n+1}{n}F_{1-\alpha}(p, n-p) \tag{4.9}$$

where n is the number of samples used to estimate μ and Σ; p the dimension of the features space; $F_{1-\alpha}$ the quantile function of Fisher distribution.

Equation 4.10 is used to determine if a component C_1 is intersecting with another component C_2, where C_1 is the component candidate to be split or to be merged with C_2:

$$(\rho-\mu)^T\Sigma^{-1}(\rho-\mu) <= \frac{(n-1)p}{n-p}\frac{n+1}{n}F_{1-\alpha}(p, n-p) \tag{4.10}$$

where μ and Σ are the mean and covariance matrices of C_1; ρ is the mean of C_2; n the number of sample in C_1.

It is worth noting that n must be strictly greater than p because both arguments of $F_{1-\alpha}$ must be strictly positive. Then, the candidate component most have more samples (n) than the number of dimensions of the feature space (p).

Split operation Algorithm 2 describes the split operation. Let C be a component, if it intersects with a component of the other class, a model candidate is computed with C split into two new components. If this candidate model has a greater *loglikelihood* than the current one, the candidate is kept. The maximization of the loglikelihood will be discussed in chapter 6. If the loglikelihood is not used the candidate is always accepted.

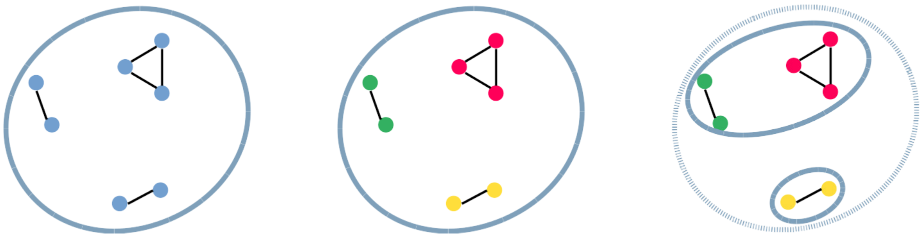
Algorithm 2 SPLIT algorithm

```
procedure SPLIT($C$,l,$M_1$, ..., $M_N$)
    $C' \leftarrow \textit{closest\_component}(C) \in M \setminus \{M_l\}$    ▷ Search the closest component from $C$ with a label $\neq l$
    if $C' \cap C \neq \emptyset$ then    ▷ If component C intersect with C'
        $C_1, C_2 = \textit{split}(C)$    ▷ See the following text for split(.)
        $\tilde{M_l} \leftarrow (M_l \setminus \{C\}) \cup \{C_1, C_2\}$
        if $\textit{llhood}(\tilde{M}) > \textit{llhood}(M)$ then    ▷ $\textit{llhood}(.)$ is the loglikelihood
            $M_l \leftarrow \tilde{M_l}$
    return $M = \cup_{l=1}^{N} M_l$
```

The split algorithm used to share the samples of the query component between two new components is the following:

Step 1: Build a graph of minimal distances between the samples of the components

Step 2: Build a set of samples per sub-graph in which all vertices are connected.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 4.5: Illustration of how the samples are shared between two new components during a split.

- If there is only one set then cancel the split.
- If there are 2 sets then go to step 3.
- If there are more than 2 sets then merged the closest sets together by average distance until having 2 sets remaining and go to step 3

Step 3: Make two new components based on the two sets of samples by computing the sample covariance and the sample mean.

This algorithm is illustrated in figure 4.5.

Merge operation Algorithm 3 describes the merge operation. Let C be a component, if it intersects with a component C' of the same class, a candidate model is computed with C and C' merged. As for the split operation, if this model candidate has a greater *loglikelihood* than the current one, the candidate is kept.

Algorithm 3 MERGE algorithm

```
procedure MERGE($C$,l,$M_1$, ..., $M_N$)
    $C' \leftarrow closest\_component(C) \in M_l$    ▷ Search the closest component from $C$ in $M_l$
    if $C \cap C' \neq \emptyset$ then    ▷ If component C intersect with C'
        $\tilde{C} \leftarrow C \cup C'$
        $\tilde{M}_l \leftarrow (M_l \setminus \ C, C') \cup \tilde{C}$
        if $llhood(\tilde{M}) > llhood(M)$ then    ▷ $llhood(.)$ is the loglikelihood
            $M_l \leftarrow \tilde{M}_l$
    return $M = \cup_{l=1}^{N} M_l$
```

4.6.2 Query Strategy

As the classifier is trained sample by sample, i.e. online, the choice of the samples is critical. A process is used to choose the next sample to explore. This process generates a distribution choice map over the supervoxels based on the prediction of the classifier.

For a pointcloud with N supervoxels extracted and $\{X_i\}_{i<N}$ the set of features of the supervoxels, the choice probability $P_c(X_i)$ of the feature X_i of the i^{th} supervoxel is defined as follows:

$$P_c(X_i) = u(X_i) * (1 - c(X_i)) \tag{4.11}$$

Where $u(.)$ is the classification uncertainty and $c(.)$ the classification confidence.

Uncertainty/Diversity As the classification is probabilistic, the output of the classifier can give a piece of information about how certain the classification is. According to equation 4.6, the closer to $\frac{1}{N}$ the probability of a sample to be part of a class, the higher the *uncertainty* is. The following equation describes how the uncertainty is computed:

$$u(X_i) = f(p(l)) \quad l = argmin_{k \in [\![1,N]\!]}(|S_k|) \tag{4.12}$$

Where, $p(l) = P(L = l|W, \Theta, X)$ and f the following function:

$$f(x) = \begin{cases} -2x(log(2x) - 1) & x >= 0.5 \\ -4x^2(log(4x^2) - 1) & x < 0.5 \end{cases} \tag{4.13}$$

Theoretically, with this definition of uncertainty, the exploration focuses in priority on uncertain areas (equation 4.13 and figure 4.6). It also tries to keep the same number of samples for each class by choosing the class with the fewest number of samples gathered (equation 4.12). Thus, this query strategy is a combination of uncertainty and diversity, as suggested by Huang and Zhou [2013]. This last feature is motivated by the fact that in most supervised learning problems, it is better to have a balanced number of samples, and on the assumption that a balanced number of samples in each class better represents the environment. Issues related to the query strategy are discussed in section 5.7.

Confidence A GMM is the sum of Gaussian distributions. The classification is supported by a mapping of the feature space made by the Gaussians functions. Thus, the probability given by an MVND may provide useful information about the structure of the dataset. Given a sample X, its classification confidence is the probability of membership to the closest component defined in 4.7. The confidence gives a measure of the dataset density. In this way, the exploration focuses on areas with less information; therefore, confidence could be interpreted as an approximation of entropy.

4.7 Conclusion

The proposed classifier has been designed to exhibit five properties: (1) ability to handle nonconvex/nonlinearly separable data, (2) ability to estimate classification uncertainty, (3) environment agnostic parameter tuning, (4) being a classifier, and

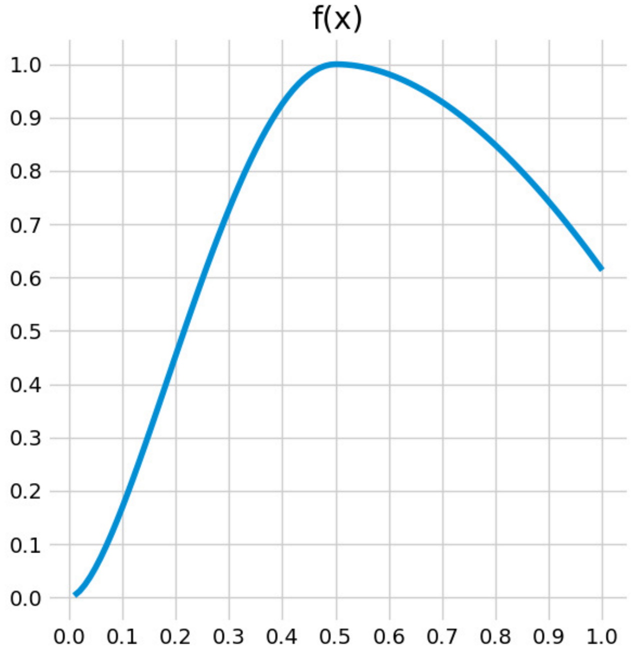


Figure 4.6: The function used for uncertainty estimation. This function gives more probability of choice to uncertain classification but also to certain classification to the chosen class, i.e the one with the fewest samples

(5) trained online. Each class is encoded by a GMM which guarantee the first, second and fourth properties. The learning algorithm is designed to process the sample one by one which fits with the fifth property. Finally, by a split and merge operations the number of components is estimated during the training. The splitting and merging are triggered by checking intersection between ellipses of tolerance of the MVND for which the sensibility is controlled by a parameter α. The independence of this parameter to different type of environments must be still tested.

CMMs is designed to be incorporated in an interactive perception framework. The proposed method is used in the next chapter to classify samples extracted from supervoxels in two classes: samples from relevant areas and samples from the background. CMMs predicts the class of the new supervoxels perceived and is used to build the relevance map. Also, CMMs is evaluated with extensive experiments in chapter 6 in which the influence of the different components of the classifier are assessed, different values of the hyperparameter α is tested on different environments.

Chapter 5
Relevance Map

The results and text of this chapter have been partially published in the following articles :

- Le Goff, L. K., Mukhtar, G., Coninx, A., and Doncieux, S. (2019). Bootstrapping Robotic Ecological Perception from a Limited Set of Hypotheses Through Interactive Perception. arXiv preprint arXiv:1901.10968.
- Le Goff, L. K., Mukhtar, G., Le Fur, P. H., and Doncieux, S. (2017, April). Segmenting objects through an autonomous agnostic exploration conducted by a robot. In *Robotic Computing (IRC), IEEE International Conference* on (pp. 284-291). IEEE.

Other contributors:

- Stéphane Doncieux, Sorbonne Université (Thesis supervisor)
- Ghanim Mukhtar, formerly, in 2017, Sorbonne Université (Engineer)
- Alexandre Coninx, Sorbonne Université (Maitre de conférence)
- Pierre-Henri Le Fur, formerly, in 2017, Sorbonne Université (Master Student)

Contents

5.1 Introduction

In this chapter, a method is presented and discussed to allow a robot to segment an environment by interacting with it. The segmentation separates parts that are moveable with a push primitive. This segmentation allows the robot to focus on relevant parts during further learning steps or when solving a task. Indeed, it is assumed that relevant parts for more complex or advanced tasks are those the robot can move with a basic action primitive. The segmented visual scene is called a *relevance map*. The proposed framework uses CMMs presented in the previous chapter. The method is developed to be applicable to most kinds of possible environments. Thus, it could be a tool to bootstrap the learning and adaptation process of the robot. The approach is summarized in figure 5.1.

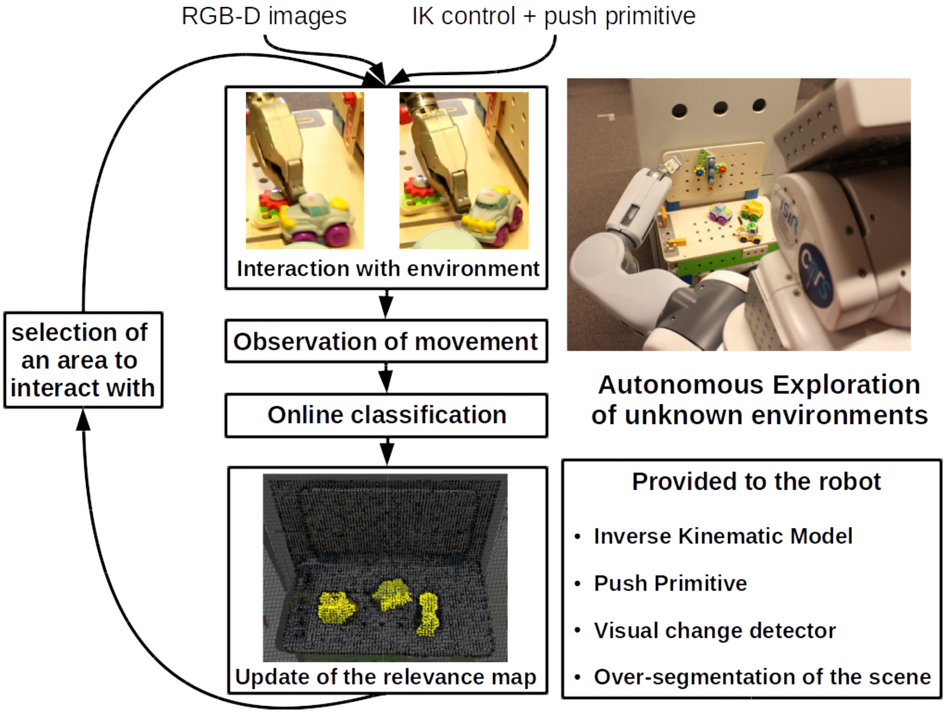

RGB-D images
IK control + push primitive
Interaction with environment
Observation of movement
Online classification
Update of the relevance map
selection of an area to interact with
Autonomous Exploration of unknown environments
Provided to the robot
• Inverse Kinematic Model
• Push Primitive
• Visual change detector
• Over-segmentation of the scene

Figure 5.1: Schema presenting the general approach presented in this chapter

This study is between two domains : interactive perception (see section 5.2) and saliency map generation (see section 5.3). When the first lets the robot interact with its environment to have a better understanding or perception, the second aims at segmenting 2D images to separate salient parts, i.e, parts that attract the attention of a human or agent, from the background. The present method aims at building a perceptual map, called *relevance map*, with the same properties as saliency map by using interactive perception framework. While saliency maps are built by studying human perception and show which areas are salient for humans, relevance maps are built according to the robot skills and thus show which areas the robot can interact with. The protocol to build such relevance maps is explained in section 5.4 and the method is evaluated by experiments explained in section 5.5. Results are presented in section 5.6.

5.2 Interactive Perception

Interactive perception is a domain of robotics in which actions are used to enhance perception and vice versa. By interacting with the environment and observing the changes, the robot is able to build a rich representation of its surrounding linked with its actions. The main claim of this field is that interactions with the environment reveal or isolate novel sensory signals (Bohg et al. [2017]). Moreover, interactions allow the robot to learn regularities in the combined space of sensory information, action, and time. These regularities are the associations made between an action and a sensory signal: when executing the same action several times the same changes in sensory signals occur. By exploiting these regularities, an agent can learn to predict sensory signals based on its action and environment properties. Finally, interactive perception allows the agent to learn the causal relationship between actions and sensory responses. The majority of the studies in interactive perceptions are with humanoid robots or arm robots with 6 or 7 degrees of freedom and with vision as sensory signals.

Early works on this topic have been conducted by Tsikos and Bajcsy [1988] in which they proposed a method to separate stacked and heaped objects thanks to interactions (like push, pick and shake) executed by a robotic system. This approach makes further image processing easier. Fifteen years later, interactive perception defined as above was studied by Metta and Fitzpatrick [2003], Fitzpatrick and Metta [2002, 2003]. In their works, a humanoid robot learns to segment a single object on a table and to recognize its arm. The robot interacts with its surrounding and uses optical flow extracted from 2D images to detect motions. By observing the motion as a consequence of its actions the system is able to segment the object from the background. In those studies, motions are restricted to planes and the experiments are tabletop scenarios in order to simplify the problem.

Most works on interactive perception start with a passive image processing step in which a first segmentation is done. This segmentation could be an over-segmentation with segments that are smaller than the objects [van Hoof et al., 2014, Schiebener et al., 2011]. In this case, assumptions are used to maximize the probability to interact with segments that are part of objects. In other approaches, segments are object candidates [Gupta and Sukhatme, 2012, Chang et al., 2012, Bergström et al., 2011, Hermans et al., 2012]. Object candidates are clusters of pixels in 2D images or clusters of points in pointclouds. The actions of a robot are then designed to reject or confirm these hypotheses. The complete interactive perception method generally follows the cycle depicted in figure 5.2 (Bohg et al.

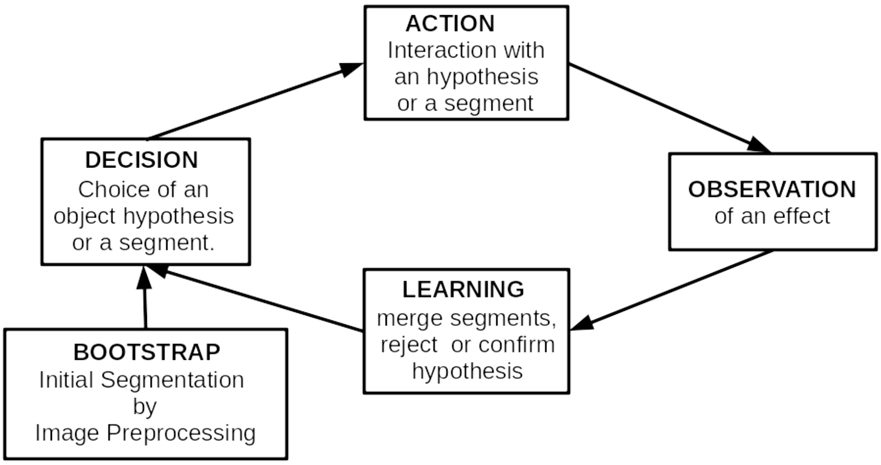

ACTION
Interaction with an hypothesis or a segment
OBSERVATION
of an effect
LEARNING
merge segments, reject or confirm hypothesis
DECISION
Choice of an object hypothesis or a segment.
BOOTSTRAP
Initial Segmentation by Image Preprocessing

Figure 5.2: General workflow of interactive perception methods.

[2017]) :

1 Choice of an object hypothesis or of an area of the visual field. In this step, a part of the environment is chosen by the system to apply an interaction. This area is often called object hypothesis or object candidate but can be a region smaller than an object when the bootstrap phase does not aim at generating object segmentation.

2 Execution of an action primitive on the chosen area. An *action primitive* is often provided to the robot like grasping, pushing, pulling, lifting, etc ...

3 Observation of the consequences of the action. This step can consist in rejecting or confirming object hypothesis or more generally detecting a change in the environment. In both cases, this step needs a *detector of change* which is often a motion detector.

4 Update of the perception according to the change in the environment and the action associated. This update can be based on a learning algorithm or on a simple heuristic.

Most works in interactive perception are focused on objects. The goals are objects segmentation, recognition, and manipulation by an autonomous exploration with a humanoid robot. As this chapter is focused on object segmentation, the following section summarizes objects segmentation by interactive perception.

5.2.1 Object Segmentation by Interactive Perception

The main goal of this set of studies on interactive perception is to obtain a clean segmentation of several objects in a cluttered environment. At the end of an experiment, the objects have to be well separated from each other. The bootstrap step often uses passive computer vision methods without any interaction. This step

Ref	Goal	Priors	Initial Segmentation
Kenney et al. [2009]	OS	RB, PM	-
Fitzpatrick and Metta [2003]	OS	TS, RB, PM, OD	-
van Hoof et al. [2014]	OS	TS, RB, AP	pixel clustering, PS
Ude et al. [2008]	OS, OR	OH, HA	-
Gupta and Sukhatme [2012]	OS, OR	TS, RB, AP	color-based clustering, PS
Chang et al. [2012]	OS, OR	TS, RB, AP	pixel clustering, PS
Hermans et al. [2012]	OS	TS, RB, AP	PS
Hausman et al. [2013]	OS, OR	TS, RB, AP	RANSAC (shape primitives)
Bersch et al. [2012]	OS, OR	TS, RB, AP	PS
Kuzmič and Ude [2010]	OS, OR	RB, AP, SP	SIFT, RANSAC
Schiebener et al. [2011]	OS, OR	RB, AP, SP, TO	Harris Corner, RANSAC, PS
Schiebener et al. [2014]	OS, OR	TS, RB, AP, SP	saliency map, difference of gaussian
Bergström et al. [2011]	OS	RB, PM, AP, TS	HSV Histograms, 3D Ellipsoids
Xu et al. [2015]	OS	RB, PM, AP, TS	Supervoxels
Eitel et al. [2017]	OS	RB, PM, AP, TS	surface-based segmentation
Our Approach	Relevance Map	AP	Supervoxels

Table 5.1: Methods in interactive perception to segment objects. The goals are object segmentation (OS) or object recognition (OR). The priors are the following : tabletop scenario (TS), rigid body (RB), action primitives (AP), planar motion of the objects (PM), object database (OD), textured objects (TO), object in hand (OH); shape primitives (SP), and human assistance (HA). In initial segmentation, PS stands for plane segmentation

introduces assumptions about objects and environments. Table 5.1 summarizes the priors and techniques used for object hypothesis generation.

The studies of van Hoof et al. [2014], Gupta and Sukhatme [2012], Chang et al. [2012], Eitel et al. [2017], Pattent et al. [2018], Chaudhary et al. [2016] have a similar goal: separating stacked or heaped objects from each other. Their studies are limited to a tabletop scenario which allows to segment the table by using most the time a random sample consensus (RANSAC Fischler and Bolles [1981]). Also, objects are assumed to be rigid bodies.

In the study of van Hoof et al. [2014] the bootstrap phase consists of an over-segmentation based on color and position of pixels which produce unicolor and convex segments. They assume that each segment (from the over-segmentation) are part of one object, thus, a true segmentation of objects is an outcome of merging those segments. Then an arm robot interacts with the scene thanks to a push primitive. The exploration is guided by maximizing information gain. Shannon entropy is approximated with Kullback Leibler divergence. The process is iterative: the robot pushes the segments until having only one segment per object. After each interaction, based on motion information the segments are merged.

The work of Chang et al. [2012] is similar but uses a grasp primitive and a push primitive called "perturbation" push for the interaction process. The robot separates objects with its "perturbation" push and then tries to grasp an isolated object. Iteratively, the objects are tidied. They work with textured and textureless objects. They use texture based keypoints for spatial matching and tracking. For textureless objects,shape-based keypoints are used.

Gupta and Sukhatme [2012] consider objects of similar types, Duplo bricks, of different colors and size. The goal is to sort the bricks by color or by size. For that the robot has three primitives: pick and place, spread action and tumble action. Tumble and spread actions have, like in work of Chang et al., to separate heaped bricks. As the bricks are unicolored, objects hypotheses are built with Euclidian distance based clustering in color space. Also, spatial Euclidian clustering is used to segment the heaps of bricks.

Eitel et al. [2017] trained a convolutional neural network to compute the probability of success of a push action to separate stacked objects. A dataset is built by letting a robot interact with the objects using push actions and an expert user assesses the success of each action to label each sample. Before the interaction step, a database of push action candidates is generated thanks to a surface-based segmentation, then the system has just to test them. All the experiments are done a tabletop setup which allows them to segment the table. This work does not aim at learning a perception as their system does not learn a representation nor a segmentation but push actions with a high success rate to separate stacked object. It aims at enhancing action thanks to perception.

Bergström et al. [2011] have a slightly different goal from the three previous studies: segmentation of unknown objects based on appearance hypothesis and rigid body hypothesis. Assuming two adjacent objects on a table, if both objects are different, passive image processing based on appearance segmentation is easy and can be checked through an interaction between the robot and the environment. If both objects are identical, the robot must interact with them to verify the number of objects. If the objects are separated after the actions of the robot then there are two objects, otherwise, there is only one object. They consider only rigid body objects. Thus, their goal is to overcome issues of object segmentation by adding interaction to computer vision techniques.

In the work of Hausman et al. [2013] the objective is to segment objects in a cluttered environment with textureless objects. To handle textureless objects, only box-like and cylinder-like objects are considered. Thus, object hypotheses are generated by matching boxes or cylinders to the RGB-D image; to finally have a set of objects hypotheses ordered into flat or round categories. Bersch et al. [2012] is a follow up of work of Hausman et al. [2013] by adding processes for textured objects by using Corner detectors and good features to track for textured objects.

Working only in tabletop environments allows a system to consider a large variety of objects but limits the environment types. The rigid body hypothesis is mandatory to build a complete object hypothesis based on motion. In other words, this assumption is: if an object hypothesis after a movement keeps the same shape and appearance then it is most likely one object. Of course, this hypothesis does not fit with articulated or soft objects but it is reasonable for most everyday objects.

To overcome the plane segmentation step, some studies make assumptions about the shapes of the objects. Candidate objects generation then relies on comparisons of objects with primitive shapes, such as cylinders, planes, or spheres or extracts shape descriptors that characterize those shapes.

Schiebener et al. [2011, 2014], Kuzmič and Ude [2010] aim at segmenting objects in cluttered environments with highly textured objects without removing the table by plane segmentation. Kuzmič and Ude [2010] consider only objects composed of planar surfaces. They use SIFT descriptors to estimate surface on 2D pictures. it is then limited to highly textured objects. Schiebener et al. [2011] used random sample consensus (RANSAC) to find planes and cylinders in a picture and generate object hypotheses. Harris corner detector is used to generate keypoints for 3D shapes estimation. Thus they also needed highly textured objects.

Schiebener et al. [2014] extends this work. The method no longer relies on local features descriptors which need textured objects but instead relies on clustering similar color and shape points from an RGB-D image. In this case, textured objects is a difficulty rather than a help. Indeed in a highly textured environment, such generation of objects hypothesis comes up with a lot of candidates. To speed up the exploration the robot considers highest objects. This heuristic allows the system to first avoid the objects candidates possibly part of the table. The exploration also puts priority into small size candidates.

These three studies do not segment planes, so could be applied in any kind of environment but assumptions are made on objects themselves, about their shapes, textures, colors. Also, all the validation experiments were conducted on tabletop scenarios.

5.2.2 Discussion

Interactive perception studies aim at enhanced objects segmentation and recognition by adding a robot interacting with the scene it perceives. These approaches aim at both discovering and separating objects which is a complex goal in robotics. A predefinition of what an object is and assumptions about environment structure is required for candidate objects generation step because it involves passive image processing. Scenarios are typically restricted to tabletop scenarios, in which objects are on a flat surface, to easily distinguish objects from the background. This is a significant limitation to the range of environments that can be handled by a robot. An alternative is to make assumptions about objects, e.g. about their shapes or their textures, but it also requires an a priori definition of the possible shapes of all objects with which the robot may interact. Finally, to prevent any assumptions about the environment or the objects, a human teacher or helper can be involved; however, this reduces the autonomy of the robot.

In the present study, the goal is to reduce the number of assumptions related to the objects and the environment to pave the way to more adaptive robot behaviors (Doncieux [2016]). Provided that the perception system actually sees the objects This limitation comes from the perception device, not from the proposed method., a single assumption is used: objects are parts of the environment that the robot can move. The robot uses interactive perception to learn to distinguish relevant objects from the background. An important feature of the proposed method is that the concept of object does not need to be defined; it relies only on the concept of relevance.

5.3 Saliency Map

Saliency is directly linked to the study of human attention. A salient part of the visual field is a part that attracts the gaze (Itti and Koch [2001]). Humans do not pay attention to all elements in a scene but only to *salient* parts, even if they are at the periphery of their visual field, the attention of a human could be attracted by something very salient (Wenzel et al. [2016]). This capability simplifies and accelerates the analysis of the environment and would then be interesting to develop on autonomous robots.

In the computer vision community, methods have been proposed to build saliency maps from a picture. A saliency map shows the distribution of salient components in the picture. It assesses which parts will most attract the visual attention of a human (Borji et al. [2014]). Saliency maps can be built by different methods. The three main methods are salient object detection (SOD), fixation prediction (FP) and object proposal generation (OPG) (Borji et al. [2014]). The aim of SOD is to generate a saliency map that represents with high accuracy the most salient object in a picture. The saliency map is composed of regions in the picture that represent salient objects. The aim of FP is to produce a saliency map that represents possible fixation points for human attention; the corresponding saliency map is a set of points. Finally, the aim of OPG is to propose bounding boxes that might include an object. The result of OPG is not exactly a saliency map, but it shares the same properties.

In the following, the focus will be on the SOD methods, which are the closest method to the one introduced later herein.

5.3.1 Salient Object Detection

SOD is used to detect the most salient object in an image and then produces a clean segmentation of object boundaries. Most methods focus on detecting one salient object (the most salient), but some attempt to detect several objects (Borji et al. [2014]). Most of the latest methods (Jiang et al. [2013b], Maleki [2017], Tan and Yan [2017], Zhu et al. [2014], Li et al. [2013], Jiang et al. [2013a], Kim et al. [2014], Yan et al. [2013]) in SOD follow almost the same workflow:

Step 1 The picture is over-segmented into regions or blocks. Blocks are rectangles on the image used to compute the visual features while regions are clusters or segments. Most of the time they use superpixel methods, in which all regions have comparable size. But region with various sizes could be used. Superpixel methods segment the image into little regions which represent clusters of similar pixels with respect to spatial and color criteria. This produces the advantage of "smart" downsampling and also allows the system to extract meaningful features, for instance, color histograms.

Step 2 Extraction and fusion of different saliency maps. This step consists in filtering or pixel-wise classification of the pictures to obtain different saliency maps based on different criteria. The filtering is based on masks which correspond to a prior about saliency. Some recent methods extract different features that are often orthogonal, to produce different maps that characterize different aspects of the saliency. The fusion of saliency maps is either heuristic or learned.

Step 3: Build the final segmentation. The outcome of the second step is a grey-scale map in which white areas represent salient regions. This step consists in binarizing the grayscale map to obtain a precise segmentation of the most salient object in the picture. To get it, they use either a fixed threshold or an adaptive threshold process.

SOD methods have strong assumptions about what makes an object the most attractive for a human. These priors are used to build masks or features. The following priors are relative to human vision or about how humans look at a scene, i.e. how humans focus their attention :

- Center prior: Salient objects are more likely to be in the center of the picture (Liu et al. [2014], Jiang and Davis [2013], Peng et al. [2013]),
- Background prior: The narrow border of the image is part of the background (Liu et al. [2014], Li et al. [2013], Jiang et al. [2013a]),
- Focusness prior: The camera often focuses on a salient object to attract attention; this can be defined as the degree of focus blur (Jiang et al. [2013c]),
- Boundary connectivity prior: Salient objects are less connected to the image border (Zou et al. [2013], Zhu et al. [2014]),
- Color prior: Certain colors seem to be more attractive to humans, e.g. salient objects are more likely to contain warm colors such as red or yellow (Shen and Wu [2012], Liu et al. [2014], Jiang and Davis [2013], Peng et al. [2013]),
- Semantic prior: Humans pay more attention to certain objects such as faces, cars, dogs, etc. (Shen and Wu [2012]).

The boundary connectivity, background, and center priors suggest that salient areas are always around the center of the image. The focusness prior assumes that the image has been taken by a human who knows where to focus. These priors cannot be used to detect objects as this would imply that a robot knows where to center its camera and therefore knows where an object is located. The color and semantic priors are specific to humans and may not be relevant in certain situations.

Other priors are heuristics in relation to what an object may look like in a 2D image:

- Objectness prior defines a measure of "objectness" based on a provided definition of what an object is and then relies on this measure to compute saliency (Jia and Han [2013], Jiang et al. [2013c])

- Spatial distribution prior: If a color is widely distributed in an image, the salient object will likely not contain this color (Jiang and Davis [2013])

These priors can be useful, but they are not necessary and can reduce the generality of the method. In the method proposed herein, most of these priors are replaced by the interaction of the robot with the environment.

Objectness prior is built by supervised learning methods. Supervised learning is mainly used with a dataset including ground truth human annotation of salient regions. Other priors could be either learned from labeled dataset or rely on engineered filters or masks.

A non-exhaustive list of the features used in SOD approaches:

- Color Histogram: histogram computed on HSV Lab or RGB domain.

- Local or global contrast.

- Background descriptor.

- Texton histogram : texture feature/descriptor.

The most widely used feature is the contrast (global or local) because of the assumption that variability in environments property is more salient than uniformity. For instance, color contrast or a sharp edge may attract human attention. Processing the contrast of features aims at determining the uniqueness of pixel, patch or region, i.e. the rarity of the region in the pictures.

According to Borji et al. [2015], the best performing methods have three features in common:

- Superpixels: contrary to block-based approaches, superpixels produce an accurate object boundary segmentation.

- Background prior: assumes that the borders of the image belong to the background. This contrast with the location prior, which assumes a specific location for a salient object in an image; usually, this is the center of the image. This assumption is strong and restricts the method to single object detection. Moreover, an autonomous robot with no concept of object will not be able to center the image around the area of interest.

- Machine learning algorithms used to train the model of saliency. Discriminative regional feature integration (Jiang et al. [2013b]) can be used to train a regression model based on a 93-dimensional features vector. This allows the method to be adaptable and scalable to more complex scenarios.

5.3.2 Discussion

These methods are used to build models of what humans would consider as salient. Furthermore, they are focused on static 2D pictures, rather than on the stream of images that a robot can collect while interacting with its environment. The focus here is not on building human-like saliency estimation. The question addressed is *what are the most relevant objects in a real scene for an agent with given capacities and with a certain goal ?* According to Borji et al. [2015], a region-based approach (i.e. superpixel) with supervised learning is an efficient method for building a saliency map. In this work, we thus propose a new region-based method to detect relevant objects based on self-supervised learning. Relevance is a concept similar to saliency, but that depends on task and robot features instead of human features.

5.4 Method

5.4.1 Overview

The goal of our method is to produce a *relevance map* through an autonomous exploration driven by a robotic arm. The robot explores an unknown, dynamic[1], environment. This exploration is driven by a relevance map of the environment, which is built online. Our approach could actually bootstrap most of the methods described in section 5.2. We follow the main principles of interactive perception and SOD. The system first over-segments the scene into regions and then classifies them to generate a grayscale map representing the relevance of the regions. It then chooses an area to explore, interacts with it, observes the effects on the environment, and updates the classifier and the relevance map (see Figure 5.3). The perception relies on an RGB-D camera (Microsoft Kinect 2[2]) to retrieve a 3D pointcloud of the scene. This camera is an active depth camera that have troubles to perceive dark and reflective surfaces (Lachat et al. [2015]). This limitation comes from the perception device, not from the proposed method.

Figure 5.3 presents the general workflow of the method. The exploration is sequential, with each iteration structured into 5 steps :

Step 1 Over-segmentation of the pointcloud into regions of the same size, called supervoxels. The over-segmentation is described in section 3.3.1. Visual features are extracted to characterize each region (see section 5.4.2).

[1] In this work, "dynamic" means that the state of the environment is not reinitialized at the beginning of each iteration.

[2] Other kind of 3D cameras could be used such as stereoscopic cameras

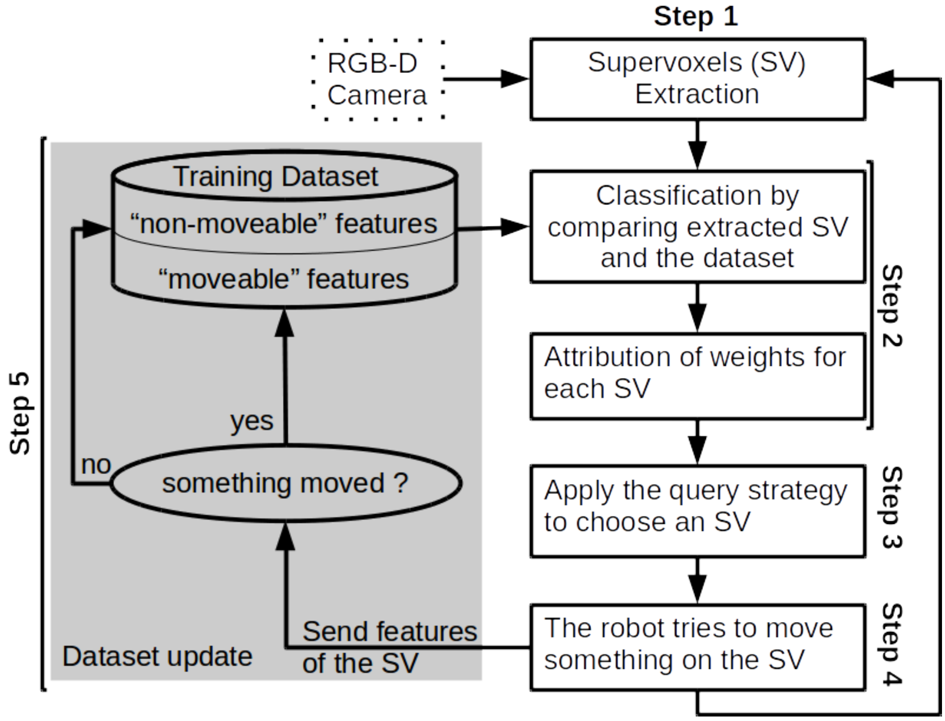

Step 1
RGB-D Camera
Supervoxels (SV) Extraction
Training Dataset
"non-moveable" features
"moveable" features
Classification by comparing extracted SV and the dataset
Attribution of weights for each SV
Step 2
yes
no
something moved ?
Step 5
Apply the query strategy to choose an SV
Step 3
Dataset update
Send features of the SV
The robot tries to move something on the SV
Step 4

Figure 5.3: General workflow of the approach.

Step 2 Computation of the *relevance map* based on the over-segmentation and according to the prediction of the classifier. This step is described in section 5.4.3.

Step 3 Choice of a supervoxel with which to interact. This choice is driven by a query strategy that relies on the relevance map (see section 5.4.4).

Step 4 The robot interacts with the selected supervoxel with its push primitive (see section 5.4.5).

Step 5 Observation of a possible effect. A basic change detection method is applied locally on the chosen supervoxel. The features of the supervoxel are stored in the database as samples. A label of 1 is used if a change is detected, and a label of 0 is used otherwise (see section 5.4.6).

In the rest of the dissertation, we call an interaction, the execution of these 5 steps.

At the beginning of the exploration, all relevance scores are initialized to 0.5. Without any information about the environment, all supervoxels are assumed to be uncertain and must be explored.

5.4.2 Features Extraction

To estimate the relevance of a supervoxel, a classifier is trained on features based on the color and shape of the supervoxels. Each feature characterizes one supervoxel.

The features used in this chapter are the following:

- Color CIELab histogram: A five-bin histogram is computed for each dimension of the CIELab color domain on the colored pointcloud of a supervoxel. Then, these three histograms are concatenated in a 15-bins histogram.

- FPFH: FPFH is a common descriptor that characterizes shape based on a pointcloud of normals (Rusu et al. [2009]). It is described in detail in 3.3.2. In this paper, FPFH is extracted from the pointcloud including the targeted supervoxel and its neighbors. The average descriptor is computed to finally obtain a 33-dimensions feature of the supervoxel.

The concatenation of the CIELab histogram and FPFH features characterizes a supervoxel. It is a vector of size 48.

5.4.3 Building the Relevance Map

Each supervoxel is weighted with a value between 0 and 1. These values represent the relevance of the supervoxel. These relevances are predictions of the classifier trained online during the exploration. They represent the probability of a supervoxel of being part of "something" moveable by the robot, i.e. an object. The relevance map is represented as a grayscale map on a 3D pointcloud segmented into supervoxels. Each supervoxel is colored between yellow (for maximum relevance) and black (for minimum relevance). The classifier is described in detail in chapter 4.

5.4.4 Query Strategy

After the computation of the relevance map, the process must select the next region of the environment to explore. Therefore, a *choice distribution map* is computed thanks to the query strategy of CMMs (see section 4.6.2). This choice distribution map represents the probability of a region to be chosen. The computation of these probabilities is based on the uncertainty/diversity and confidence of the classifier. The exploration is motivated by a reduction of uncertainty or an increase in the level of information (i.e. the entropy) about the environment. The computation of the choice distribution map is described in detail in section 4.6.2.

5.4.5 Push Primitive

To enable interaction with the chosen supervoxel, a push primitive is used. This primitive is divided into three steps: approach movement, straight line motion towards the supervoxel center for interaction, and reverse motion. A planning algorithm with obstacle avoidance is used for the approach phase [Şucan et al., 2012, Şucan and Chitta, 2018].

In this phase, the end-effector of the robot moves to an approach position near the target position, which is the supervoxel center. An approach position is randomly chosen among those associated with a valid motion plan, i.e. a motion plan without self-collisions and collisions with the scene.

At the end of this phase, the end-effector is at an approach point and positioned towards the target. Then, the end-effector moves to the target following a straight line of 5 centimeters and attempts to pursue this trajectory for a further few centimeters[3]. In other words, the robotic arm tries to push the target. Finally, the robot arm returns to its home position by following the reverse trajectory. How far the robot try to go further determines the strength of push. For the experiments of this chapter, this distance is fixed at 2 centimeters.

The pushing motions can take different orientations of the end-effector relative to the target before pushing it. The orientation of the gripper is chosen randomly in a cone of $\frac{\pi}{2}$ centered around the average normal of the targeted supervoxels.

5.4.6 Change Detection

As the exploration is sequential, change detection is simply a comparison between the pointcloud before and after the interaction. The comparison follows these steps:

- The *octree pointcloud change detector* method provided in PCL (Rusu and Cousins [2011]) is used to subtract the initial pointcloud (before the interaction) from the current pointcloud. This operation produces a pointcloud limited to the difference between both pointclouds.

- With a *statistical outlier removal*, provided in PCL, the noise in the difference pointcloud is reduced.

- Finally, points only in the selected supervoxel are compared with the difference pointcloud, using an *iterative closest point* algorithm implemented in PCL, to determine if this group of points is in the difference pointcloud. If this is the case, it is considered that the action of the robot produced a movement in the environment, and the feature of the supervoxel is given a "moveable" label; otherwise, it is given a "non-moveable" label.

Figure 5.4 shows a visualization of how the change detector works. The right picture represents a part of a scene before a push and the left picture after a push. The red dot represents the center of the supervoxel targeted by the system. The white areas represent the differences detected between both images. If the target supervoxel is included in the white areas, then a change is detected. The change detector considers only the targeted supervoxel, i.e. a small area around the target.

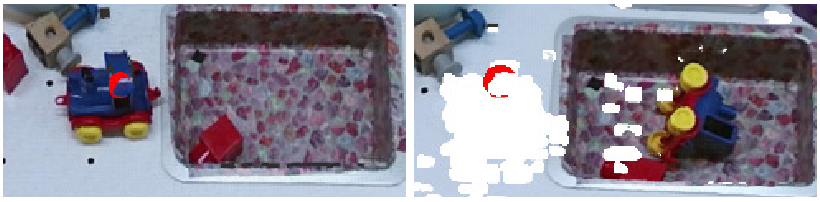


Figure 5.4: Visualisation of the change detector. The right picture represents a part of a scene before a push and the left picture after a push. The red dot on both pictures represents the target of the push primitive which is here the upper part of the blue toy. This target corresponds to the center of a supervoxel. The white areas represent the parts detected as different between both images.

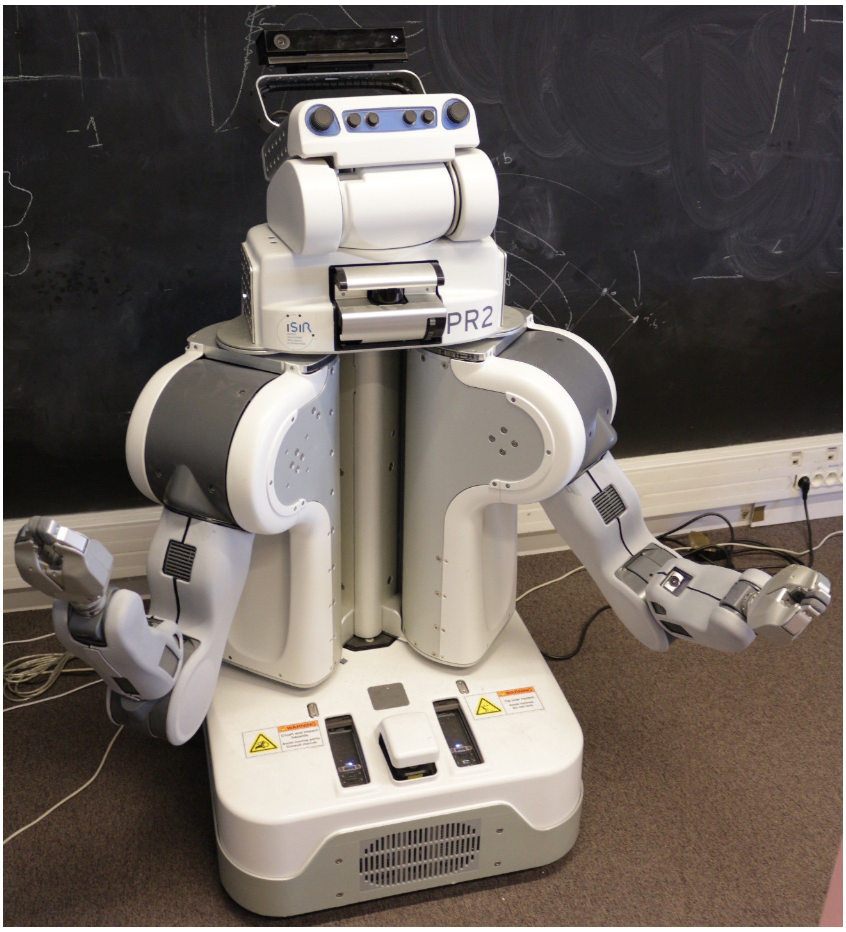


Figure 5.5: The PR2 robot is used for all the experiments presented in this dissertation. It have a wheeled base, two 7 degrees-of-freedom arms and a head mounted Kinect v2. During the experiment the both arms are used and the kinect is used as visual system. The robot remains at the same place during the whole experiment.

5.5 Experiments

5.5.1 Protocol

The experiments performed to validate the method are of two types: the ones in a simplified setup and the ones using a real robotic platform using the PR2 robot represented in the figure 5.5. The setups used for the experiments are described in table 5.2. All experiments have a fixed budget of interactions. Each experiment has a single fixed background and a set of mobile objects.

An expert is used to evaluate the quality of the classifier trained during an experiment. The expert is built by saving the pointcloud of the background without the objects at the beginning of the experiment; thus, the objects are easily separated from the background. To determine whether a supervoxel is part of the background, the points of the supervoxel are simply compared with the saved background pointcloud. This defines the ground truth of the classification, which is, of course, not known by the classifier.

During an experiment, quality measures (described in section 5.5.2) are computed after each interaction by comparing the relevance values attributed by the classifier to the supervoxels extracted from the current scene with the relevance values attributed by the ground truth.

All experiments begin with a bootstrap phase during which the choice of the supervoxels to explore follows a random uniform distribution. This phase is arbitrarily fixed to a budget of 10 iterations with which to initialize the dataset independently from the classifier which initially has insufficient data to produce reliable predictions.

For all experiments,the parameter α which controls the intersection sensibility is fixed to 0.25 (see sections 3.2.2 and 4.6.1).

Simplified Setups The experiments conducted using the simplified setups are used to evaluate the method in an ideal case. We use the gazebo simulator without any robot and with a simulated kinect. The interactions are fake, i.e. there is no robot that interacts with the environment. The class (moveable or not) of the explored supervoxel is assessed by an expert that knows in advance which supervoxels are part of an object and which supervoxels are part of the background (described in the previous paragraph). In the simplified setups, there is then no noise and no mislabeled samples. The results, in this case, are an upper bound of what can be expected in reality.

The experiments are conducted in several environments (see figure 5.6 and table 5.2). They are designed to have increasing difficulty with increasing number of shared features between the objects and the background. For instance, the setup **MoveableBalls** (5.6a) is very simple because the background is a wooden table, a flat surface with colors between orange and yellow, whereas the moving objects

[3] This control sequence is open-loop, the system updates its perception of the environment only after the execution of the control program.

Name	Fixed parts	Moveable Parts	Type	Complexity
MoveableBalls (5.6a)	A wooden table	Blue and green balls	Simplified	*
MoveableBricks1 (5.6b)	A wooden table	Red, orange, and yellow boxes	Simplified	**
MoveableBricks2 (5.6c)	A wooden table with fixed green and blue balls	Red, orange, and yellow boxes	Simplified	***
WhiteMoveableBalls (5.6d)	A white table with fixed white boxes	White balls	Simplified	***
WhiteMoveableBricks (5.6e)	A white table with fixed white balls	White boxes	Simplified	****
SimKitchen (5.6f)	A green marble kitchen counter with a double grey sink	An orange bowl, a blue cup, a black teapot, and a yellow/green spray cleaner	Simplified	***
Simple Workbench (5.7a)	A wooden toy workbench	Three multicolor toy cars	Real	***
Second Workbench (5.7b)	A wooden toy workbench with fixed wooden cube, green toy and blue and red slab	Three multicolor toy cars	Real	****

Table 5.2: Description of the setups used for the experiment of this chapter. The descriptions of the environments is separated between the fixed part and the part the robot can move. The type corresponds either to the simplified setup or to the real robotic platform. The complexity is determined qualitatively before running the experiments.

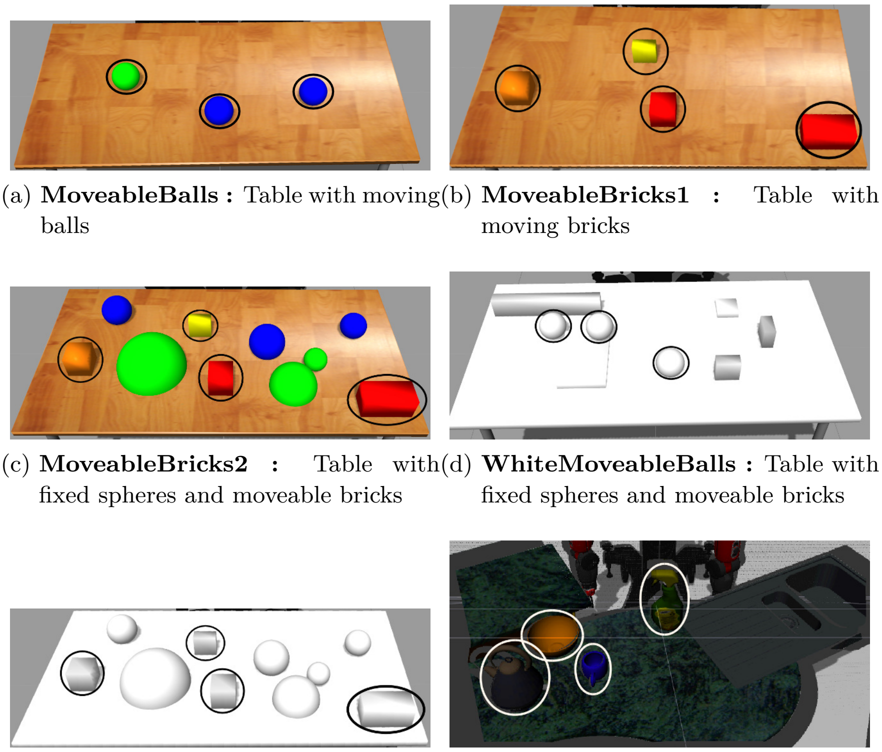
are blue and green spheres. Thus, the feature space is very easy to split. While the setup **WhiteMoveableBricks** (5.6e) is composed of a white table with fixed white spheres and moveable white boxes. In this setup, the moveable and non-moveable regions shared a lot of visual features such as the color (as the whole environment is white colored) or the shape. These criteria of difficulty are only qualitative based on what is expected to be a hard problem for our proposed classifier CMMs. At each iteration during the experiment, each object is spawned in a random position and with a random orientation.

The objects chosen for these experiments are very simple (cubes and spheres) to simplify the analysis of results according to the feature space. The experiments performed using the real robot use more complex object shapes.

Real World Experiments Experiments are also conducted with a real robot to evaluate the method in a realistic scenario. The PR2 robot is used with a Kinect version 2 sensor. In figure 5.7, the setups used for the experiments are depicted. It is based on a modular workbench toy in two different configurations. These environments are colorful and have complex shapes that allow us to test the method on a complex and realistic setup. In these experiments, the robot interacts with the environment and the classifier learns from the label produces by these interactions.

5.5.2 Classification Quality Measures

Precision, Recall, and Accuracy To measure the performance of the method, precision, recall, and accuracy are used. These are classical measures used in computer vision and more generally in classification tasks. In particular, these measures are used in most studies on SOD (Borji et al. [2015]). The following equations define



(a) **MoveableBalls :** Table with moving balls

(b) **MoveableBricks1 :** Table with moving bricks

(c) **MoveableBricks2 :** Table with fixed spheres and moveable bricks

(d) **WhiteMoveableBalls :** Table with fixed spheres and moveable bricks

(e) **WhiteMoveableBricks :** Table with fixed spheres and moveable bricks all white

(f) **SimKitchen :** A kitchen with a teapot, a bawl, a cup and a spray cleaner

Figure 5.6: Experimental simplified setups ordered by increasing difficulty. The moveable objects are marked by black and white circles.

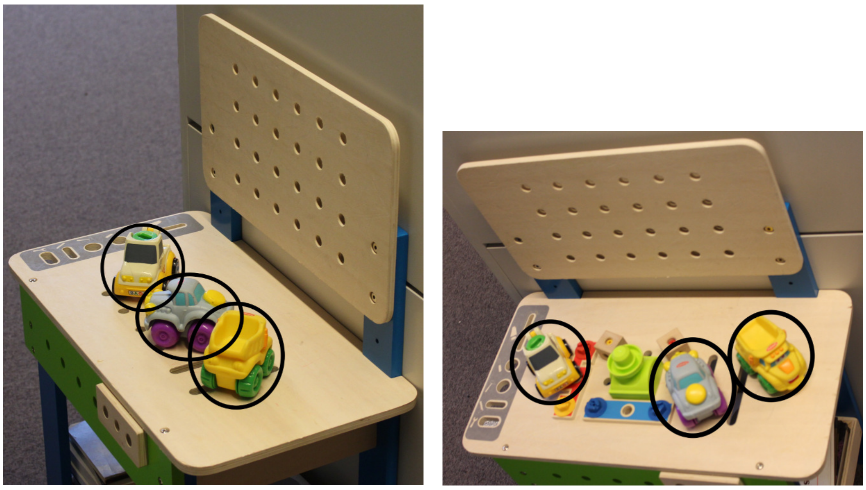


Figure 5.7: Experimental setups with the real robot ordered by increasing difficulty. The moveable objects are marked by black circles.

precision, recall, and accuracy, as used in this study:

$$\begin{aligned} precision &= \frac{tp}{tp + fp} \\ recall &= \frac{tp}{tp + fn} \\ accuracy &= \frac{1}{2}(\frac{tp}{G_{obj}} + \frac{tn}{G_{back}}) \end{aligned} \tag{5.1}$$

Where tp is the number of true positives and tn is the number of true negatives (i.e. supervoxels well classified as part of moveable objects or as part of the background, respectively); fp are false positives, i.e. supervoxels misclassify as moveable and fn are false negatives, i.e. supervoxels misclassify as non-moveable; and G_{obj} is the ground truth for parts of the environment that are objects and G_{back} is the ground truth for parts of the environment that are fixed. Their definitions, for N supervoxels extracted on a pointcloud, are as follows:

$$\begin{aligned} tp &= \sum_{i=0}^{N} P(L = 1|W, \Theta, x_i) * (1 - \delta_i) \\ tn &= \sum_{i=0}^{N} P(L = 0|W, \Theta, x_i) * \delta_i \\ fp &= \sum_{i=0}^{N} P(L = 1|W, \Theta, x_i) * \delta_i \\ fn &= \sum_{i=0}^{N} P(L = 0|W, \Theta, x_i) * (1 - \delta_i) \\ G_{obj} &= \sum_{i=0}^{N} 1 - \delta_i \\ G_{back} &= \sum_{i=0}^{N} \delta_i \end{aligned} \tag{5.2}$$

Where δ_i is the Kronecker symbol equal to 1 if the ith supervoxel is part of the background, and otherwise equal to 0; x_i represents the features of the ith supervoxel.

Measure of the Exploration Dynamic We also measure the number of samples gathered during the exploration of each class. This allows us to assess how the exploration is conducted according to the knowledge available. The number of components is also monitored to determine whether it increases with the complexity of an environment. For the experiments with the real robot, the number of mislabeled samples, which correspond to failed interactions, is counted.

5.6 Results

5.6.1 Simplified Setups

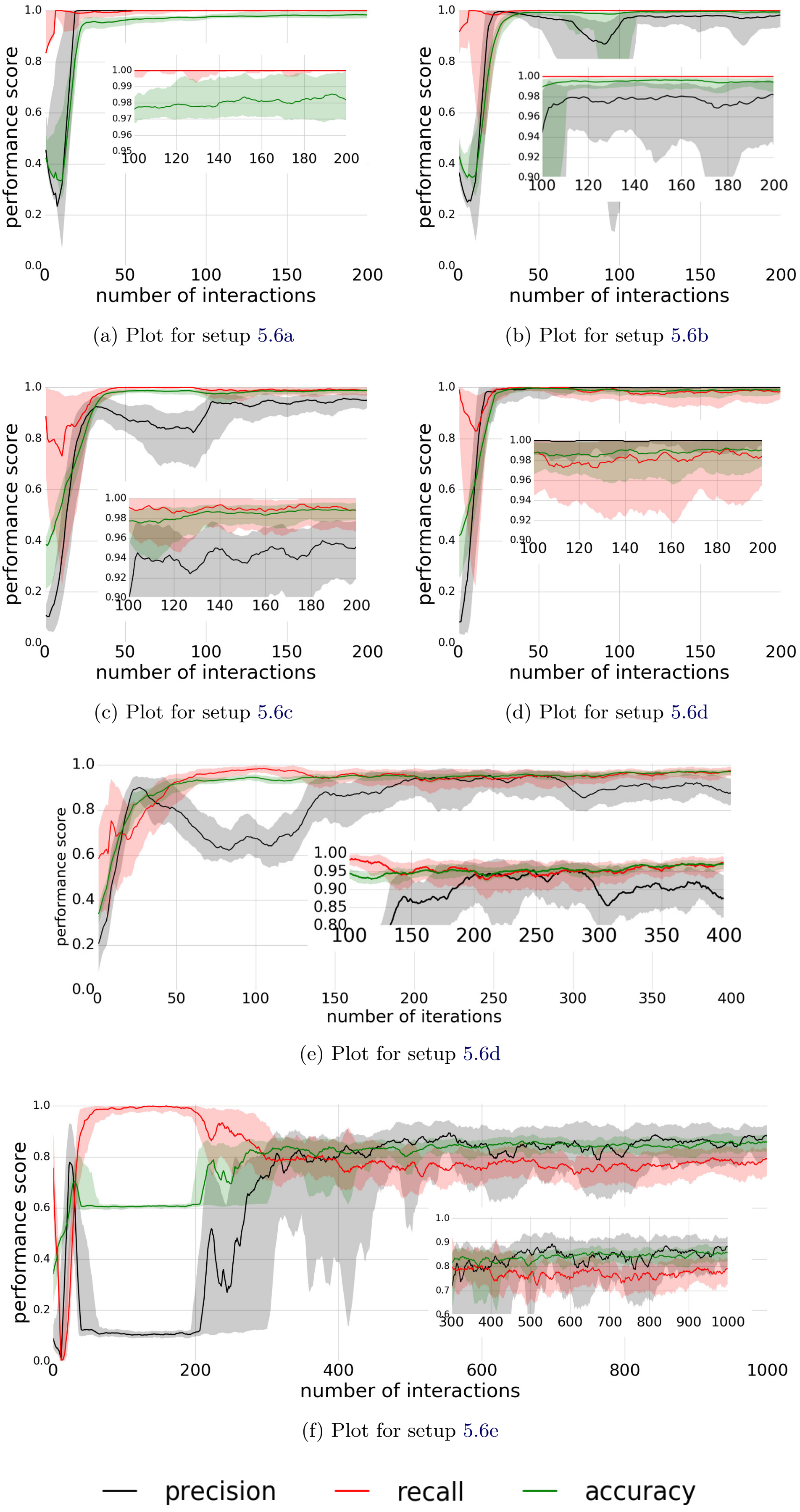
As expected, the classification reaches scores of almost 1 for **MoveableBalls** setup (5.6a) (see figures 5.8a). There is nothing complex about these setups as the moveable objects share no features with the background. However, in **MoveableBricks1** (5.6b) and **MoveableBricks2** (5.6c) setups, the performance does not directly reach maximum performances (see figures 5.8b and 5.8c). This is due to the similarities between the table and the cubes in terms of color and shape. Moreover, as the classifier takes more time to converge, the query strategy is less efficient at choosing suitable samples at each iteration, therefore, the system takes more time to gather a representative dataset as shown in figure 5.9. A minimum number of samples is needed to start splitting components because of the constraint introduced by the component intersection criterion (see equation 4.10) which explains the decrease in performance until iteration 100.

The shape feature is sufficient for setup **WhiteMoveableBalls** (5.6d) as the classification reaches scores of almost 1 (see 5.8d). For setup **WhiteMoveableBricks** (5.6e), the classification does not reach maximum performance, but this is expected as the background and moveable objects have similar shapes (see 5.8f). Even in this setup, the accuracy converges to a value above 0.8 and this seems sufficient to perform a relevant segmentation, as can be seen in figure 5.16a which depicts a relevance map obtained using setup **WhiteMoveableBricks** (5.6e). The segmentation is not perfect, but it is accurate enough to identify object hypotheses that can be validated in the next step of the robot developmental process (see section 5.2). On setup **SimKitchen** (5.6f), the performance reaches a value above 0.8 after 150 interactions (see figure 5.8e). It is better than the results for setup **WhiteMoveableBricks** (5.6e), probability because the simulated kitchen (5.6f) is richer in shapes and colors.

The query strategy is far from a uniform random sampling, which would mostly explore the background. The comparison of the numbers of samples in each class can be considered as a necessary convergence criterion: if the dataset converges to the same number of samples in both classes, it means that the classifier is able to distinguish the two classes with sufficient accuracy. But this criterion is not sufficient. If the recall score is too low, a lot of false negatives are present. As there is a higher probability to choose a supervoxel of the background, this is sufficient to have just a few true positives to reach a balanced dataset. As shown in figure 5.9, all explorations reach the same number of samples in each class with no variability over the replications, indicating the stability of the query strategy.

According to these results, setup **MoveableBricks2** (5.6c) is actually harder for the classifier to deal with than setup **WhiteMoveableBalls** (5.6d).

Figure 5.10 shows the number of components for each class at each iteration. The increasing number of components when the setup becomes more complex is worth to look at. Most likely this corresponds to cases when both classes share



(a) Plot for setup 5.6a

(b) Plot for setup 5.6b

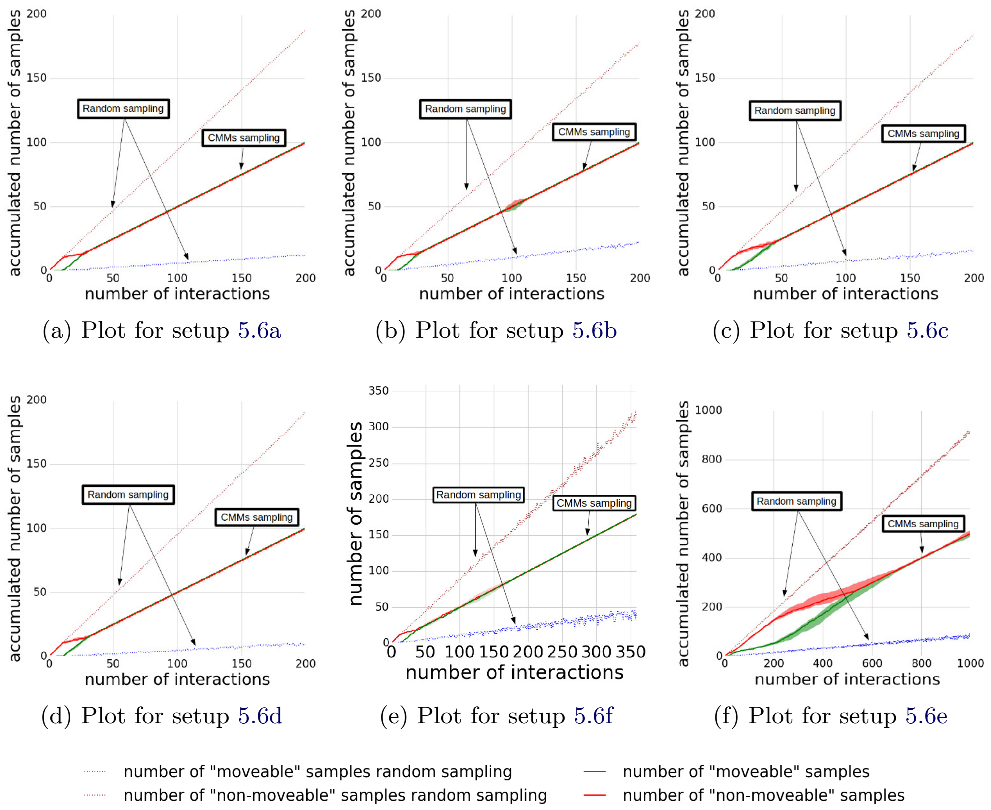
(c) Plot for setup 5.6c

(d) Plot for setup 5.6d

(e) Plot for setup 5.6d

(f) Plot for setup 5.6e

Figure 5.8: Plots of precision, recall, and accuracy for each setup presented in figure 5.6



(a) Plot for setup 5.6a

(b) Plot for setup 5.6b

(c) Plot for setup 5.6c

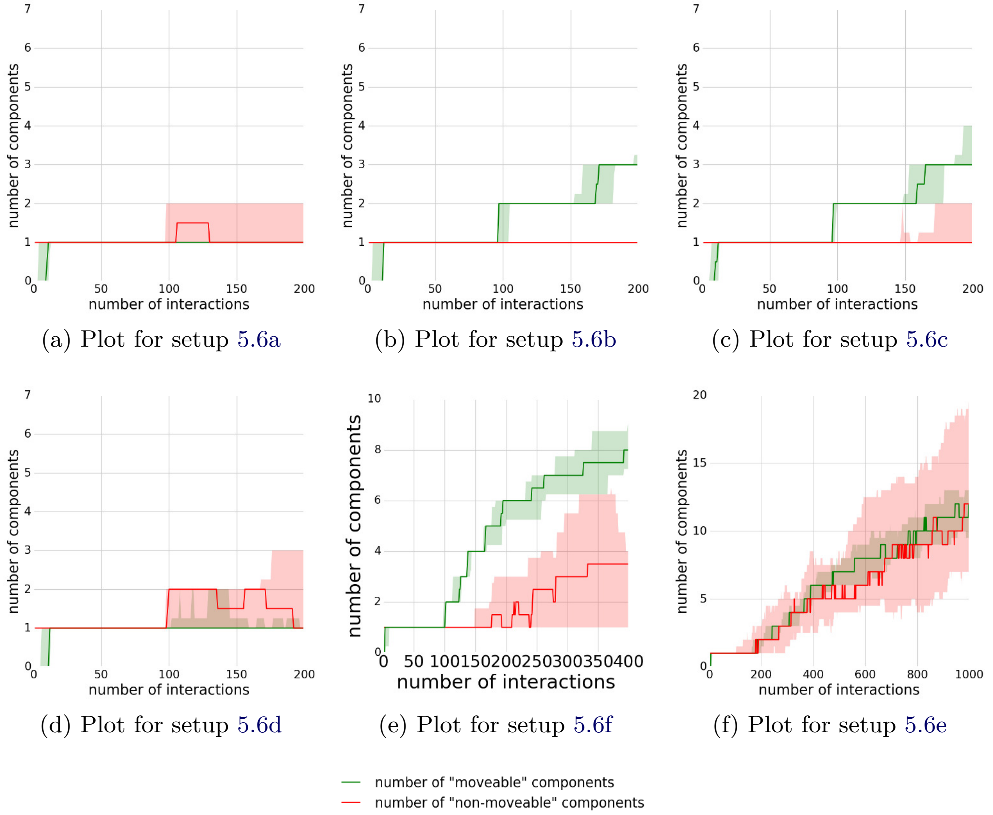
(d) Plot for setup 5.6d

(e) Plot for setup 5.6f

(f) Plot for setup 5.6e

Figure 5.9: Plots of the number of samples gathered for each class at each iteration during the experiments for each simplified setup presented in figure 5.6.

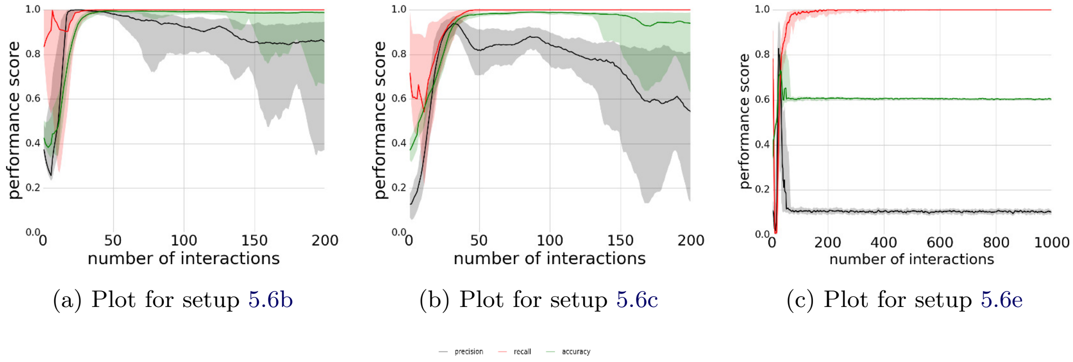
common features. Setup **WhiteMoveableBricks** (5.6e), which led to the worst results (see figure 5.8f), has a rapidly increasing number of components and does not reach a plateau within 1000 iterations (see figure 5.10f). Also, for all setups except setup 5.6e, new components are created only after iteration 100; this is a consequence of the constraint introduced by the intersection criterion (see equation 4.10). For the setup on the simulated kitchen (5.6f), the number of components are also increasing but seems to slow down. Moreover, the number of components for the non-moveable category is much lower than for the moveable category. This suggests that the background is less complex than the objects, which seems to be the case (see figure 5.6f).



(a) Plot for setup 5.6a (b) Plot for setup 5.6b (c) Plot for setup 5.6c

(d) Plot for setup 5.6d (e) Plot for setup 5.6f (f) Plot for setup 5.6e

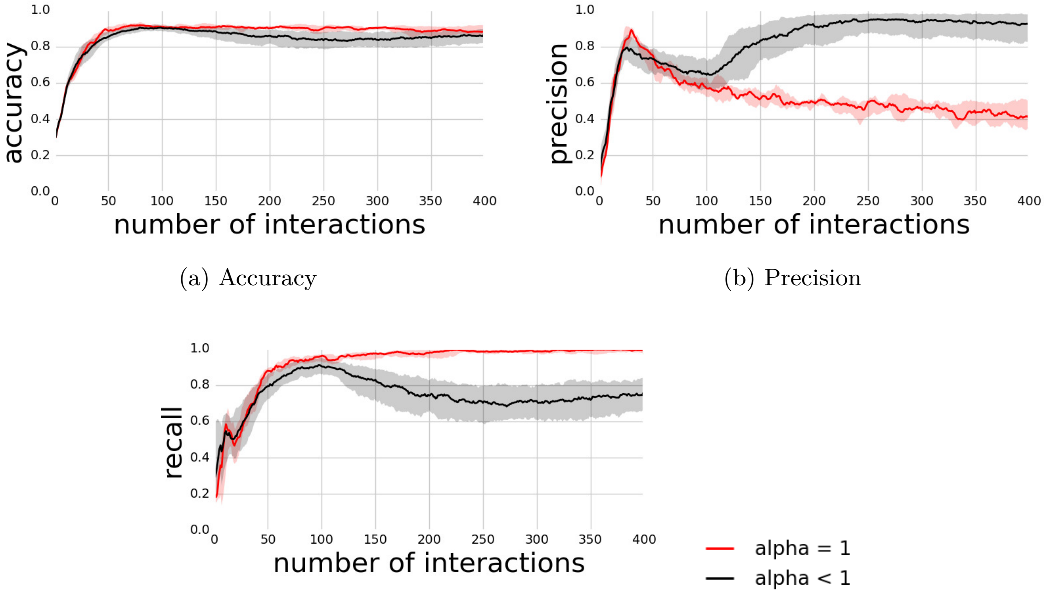
Figure 5.10: Plots of the number of components of each class at each iteration during the experiments for each simplified setup presented in figure 5.6.



(a) Plot for setup 5.6b (b) Plot for setup 5.6c (c) Plot for setup 5.6e

Figure 5.11: Plots of precision, recall, and accuracy for setups 5.6b, 5.6c and 5.6e for experiments conducted with one component per model. In these experiments, no split or merge operations were applied. These experiments are made to control the contribution of split and merge operations.

Figure 5.11 shows the performance of experiments conducted with only one component per model, i.e. where no split or merge operations are applied. In this case, performances are poorer than those including split and merge operations. This indicates that setups 5.6b, 5.6c and 5.6e involve non-convex classes and non-linearly separable datasets and justifies the need for the proposed split and merge operations.



(a) Accuracy

(b) Precision

(c) Recall

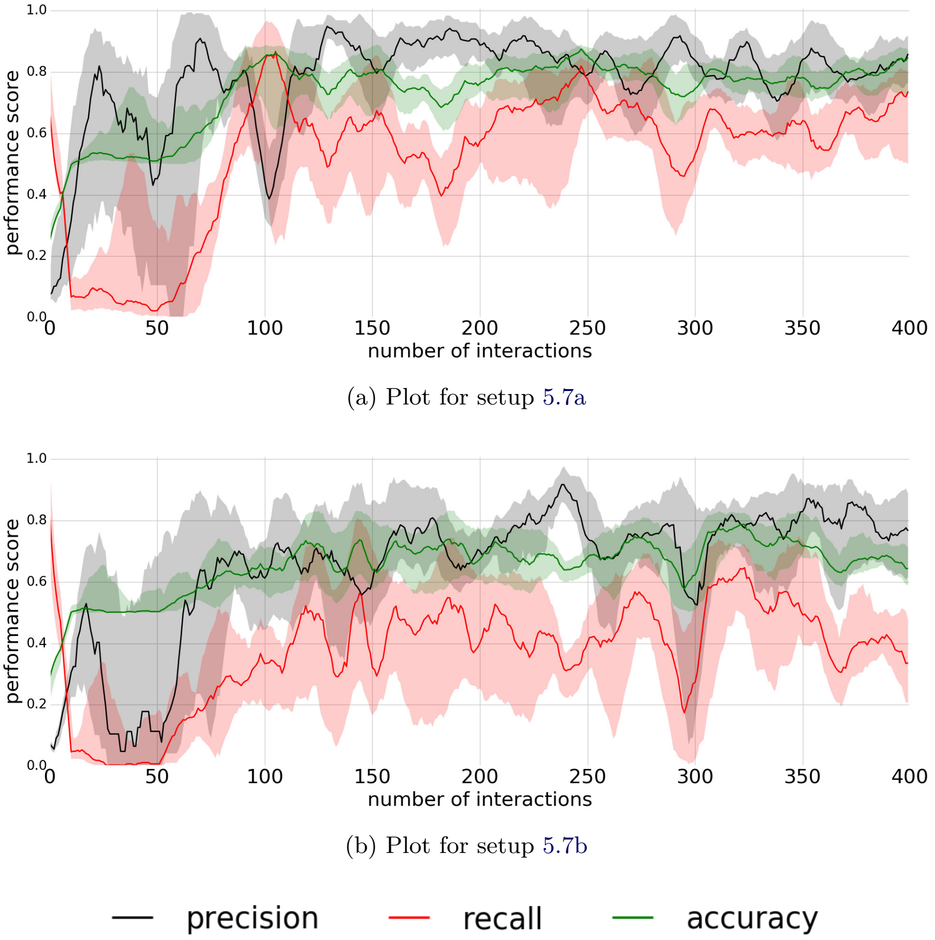
Figure 5.12: Results of the experiments conducted on the simulated kitchen with α varying between 0 and 1 with a step of 0.1. The replication with α strictly less than 1 are grouped into the black curve and equal to one into the red curve

Figure 5.12 represents the results of experiments conducted on the simulated kitchen (5.6f) with α varying between 0 and 1 with a step of 0.1. For better clarity, the replications with alphas between 0.9 and 0 are grouped in the black curve, thus, the black curve gather 100 replications and the red one 10 replications. The replications are grouped like that because there is no splits and merges only for α equal to 1. The accuracy (see figure 5.12a) is approximately the same for any value of alpha. For α strictly less than 1, the precision converged to a value around 0.9 (see figure 5.12b) and for recall to a value around 0.8 (see figure 5.12c), while for α equal to 1 the precision keep decreasing and the recall converged quickly to a value close from 1. Also, for α strictly less than 1, recalls and precisions have a low variability. The results seem to indicate that with split and merge, for any value of α (strictly less than 1), the classification reaches a sufficient quality with a bit low recall, while without split and merge (α equal to 1), the classification has a very low precision despite a high accuracy and recall.

Finally, the α parameters could be fixed to any value between 0.9 and 0. And the split and merge operations allow the system to have a better precision in classification with a lower the recall.

5.6.2 Real World Experiments

In this setup, the data is obtained after the application of the push movement primitive and the detection of an eventual change in the targeted area. As expected, the performance in the real environment does not reach the maximum level, but it



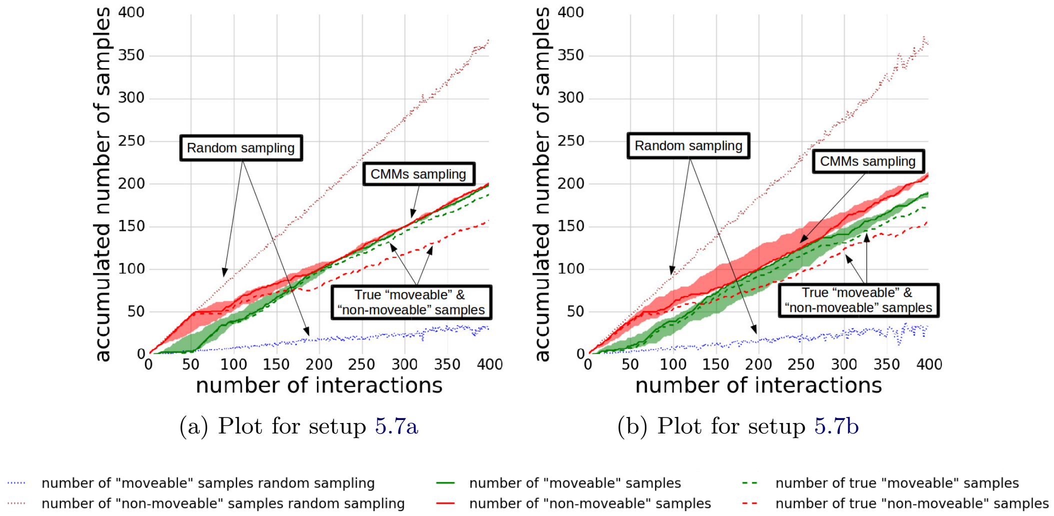
(a) Plot for setup 5.7a

(b) Plot for setup 5.7b

Figure 5.13: Plots of precision, recall, and accuracy for each setup presented in figure 5.7

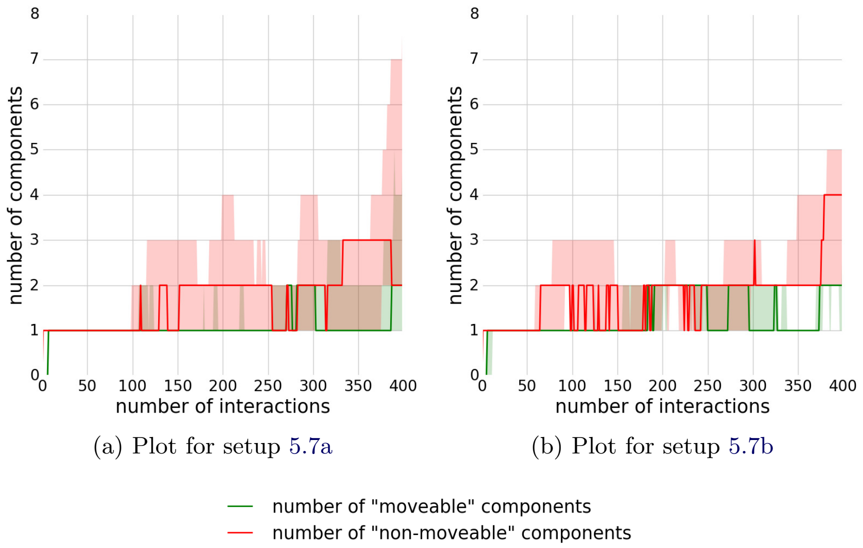
does reach 0.8 for accuracy (see figure 5.13). This is enough to produce a useful segmentation of the scene, as shown in figures 5.16b and 5.16c. The classification is less efficient for setup **Workbench2** (5.7b) than for setup **Workbench1** (5.7a), and it is less stable over the iterations. This is expected as setup **Workbench2** (5.7b) is more complex than setup **Workbench1** (5.7a).

A total of 400 iterations is enough to achieve almost the same amount of samples in both classes, as shown in figure 5.14. Compared with the simulation, the exploration produces mislabeled samples due to failed interactions and failures in the detection of motion. As the classifier is online, the training is more sensitive to mislabeled samples which introduces instability over the iterations and variability over the replications.



(a) Plot for setup 5.7a (b) Plot for setup 5.7b

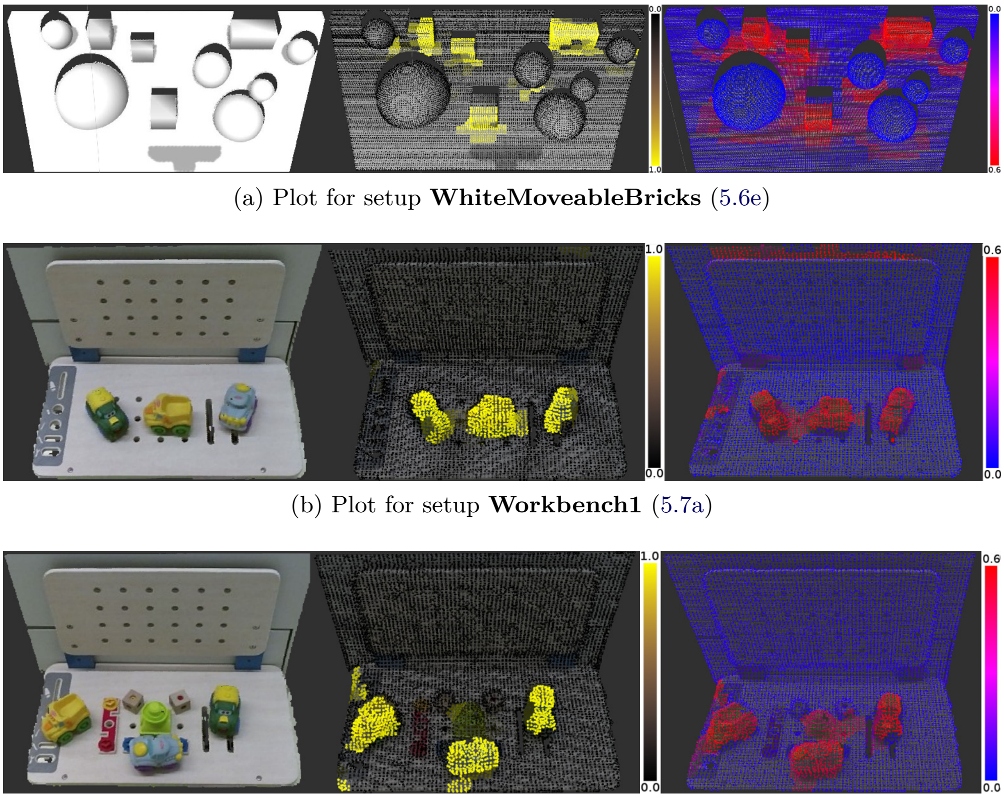
Figure 5.14: Plots of the number of samples gathered for each class at each iteration during the experiments for each setup presented in figures 5.7.



(a) Plot for setup 5.7a (b) Plot for setup 5.7b

Figure 5.15: Plots of the number of components of each class at each iteration during the experiments for each setup presented in figures 5.7.

The increasing number of components seen in the simulation is related to the complexity of the environment; with the real robot, it is also related to mislabeled samples. Indeed, mislabeled samples introduce a higher complexity in the distribution of samples in the feature space, therefore components split is more likely to occur. This explains the large variability in the number of components on figure 5.15a.



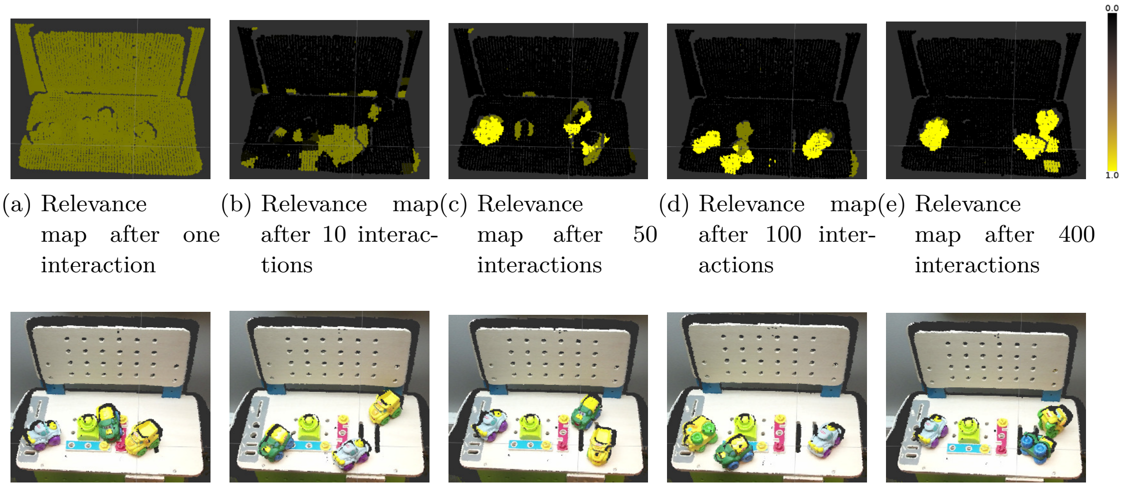
(a) Plot for setup **WhiteMoveableBricks** (5.6e)

(b) Plot for setup **Workbench1** (5.7a)

(c) Plot for setup **Workbench2** (5.7b)

Figure 5.16: From right to left: pictures of pointcloud; relevance map with in yellow the highest probability to be moveable and in black the lowest (from 0.0 to 1.0); average choice distribution map over an exploration with in red most queried areas and blue least queried areas (from 0.0 to 0.6).

Figure 5.16 presents the best performing relevance map for both real setups 5.7) and for the most complex simulated setup, **WhiteMoveableBricks** (5.6e). It also shows an average choice distribution map over all iterations, which represents the most queried areas for exploration. This map provides an insight into which parts of the environment are most considered during exploration. For the three setups, the exploration is more focused on complex areas such as moveable or fixed objects (that are then part of the background). With an exploration driven by uncertainty, this is an expected feature as the complex areas are the slowest to decrease their uncertainty. Finally, figure 5.17 shows a sequence of relevance maps generated from classifiers taken at different moments of the exploration on the setup **Workbench2** (5.7b). The first relevance map after 1 interaction is totally neutral, showing that



(a) Relevance map after one interaction

(b) Relevance map after 10 interactions

(c) Relevance map after 50 interactions

(d) Relevance map after 100 interactions

(e) Relevance map after 400 interactions

Figure 5.17: A sequence of pointclouds representing a relevance map at different points during the exploration. The bottom pictures correspond to the pointcloud used to generate each relevance maps. These images have been generated after an exploration.

all the environment is considered as moveable with a probability of $\frac{1}{2}$ while the last relevance map attributes a high probability to almost only the cars.

5.7 Discussion and Future work

The query strategy is the most crucial component of this method. On the one hand, the classifier requires a representative dataset of the scene; on the other hand, it requires a suitable sample at each iteration to learn efficiently. With only a uniform random query strategy, the dataset would be composed of an overwhelming majority of background samples; therefore, the classifier would have difficulties in converging (see figures 5.9 and 5.14). The proposed approach to drive the query strategy is an uncertainty reduction. Uncertainty is measured based on the probability of membership to each class being close to 0.5, i.e. at the border of both Gaussian mixture models. This focuses exploration on unknown or poorly known areas. The confidence of the classification is used to focus the exploration on areas in which the dataset has a low density. Confidence draws inspiration from entropy while being less costly to compute. The entropy of models is a measure of information quantity. The query strategy thus increases the representativeness of the dataset. Combining uncertainty and confidence allows the exploration process to focus on unknown and informative areas, as shown in the left picture of figure 5.16. Finally, to balance the dataset between the two classes, priority is given to query samples from the less represented class of the dataset.

The quality of the relevance map depends on the precision of the change detector. In this work, the change detector is a simple frames comparison between before and after the execution of the push primitive. This component of the framework could be easily enhanced by adding haptic sensors on the robots end-effector or real-time tracking system for motion detection. The other components are mostly independent of the change detector, thus, it can be changed without major issues.

The online training of CMMs does not offer a precise measure of convergence. In batch learning, test steps give a measure of overfitting and learning progress, which is not achievable in online learning. In online learning, establishing a test dataset is complicated as the test dataset must be sufficiently different from the training dataset to detect overfitting. Within the budget fixed for the experiment, the classifier converges, as shown in figures 5.8 and 5.13, in which precision, recall, and accuracy converge to a mean value. In addition, the exploration always reaches a balanced dataset, as shown in figures 5.9 and 5.14, suggesting convergence of the classifier. But, for the most complex setup 5.7b, the performance is unstable and decrease several times. Thus, accumulating more samples does not guarantee an increase in performance. In the next chapter (Chapter 6), an extensive study of the proposed framework is conducted to have a better understanding of the problem and, an upgrade of some parts of the model is proposed.

The classifier used in this paper is designed to be non-specific to a particular kind of environments. In particular, the hyperparameters of the model should be the same for all environments. CMMs have one hyperparameter: the tolerance ellipse size (α) which determines the sensitivity of the *merge* and *split* operations. This parameter must be tuned to have the best classification efficiency. In this study, α was fixed to the same value for all the experiments ($\alpha = 0.25$) (see sections 4.6 and 5.5.1). Thus, a compromise for a large set of environments can thus be found for the values of this hyperparameter. Moreover, varying the value of α between 0.9 and 0 does not introduce a high variability in classification quality.

The quality of the classification is conditioned by the features used. A complex environment would require the use of features that can capture this complexity to generate an efficient segmentation. However, as the feature extractor is designed prior to exploration, it could potentially reduce the kinds of environment to which the robot can adapt. The results of setups 5.6d and 5.6e show that the feature space can be more complex than required in practice (in these setups, the color descriptor is ignored by the method). As in the work of Jiang et al. [2013b], the feature space is 93-dimensional, which is more than what is actually required for a lot of problems but who can do more, can do less. In the proposed approach, the feature used is 48-dimensional, which is already large and integrates rich shape and color information. It allows the method to be more adaptive. Thus, to deal with any situation the robot may encounter, the use of a descriptor as rich as possible is recommended.

The interactive perception paradigm has a strong link with affordances. An affordance is a relational property which emerges from the agent-environment system . This concept allows to formalize a representation of the environment through the action of the robot. As the relevance map is built thanks to the interaction of the robot with a push primitive, it represents the probability of environment parts to be "pushable", in other words, it represents areas of the environment that afford the push action for the robot. As for the change detector, the primitive is an independent component of the framework, so, it is possible to change the primitive with a minimum of efforts. Other experiments with primitives like lifting, pulling, or grasping could be conducted with the proposed methods. Of course, a change detector adapted to the new primitive needs to be provided. In these cases, the relevance map would represent affordances like "liftable", "pullable" or "graspable". Chapter 7 presents an extension of the methods in this direction.

5.8 Conclusion

A method has been introduced that allows a robot to segment a visual scene into two different classes: regions that belong to *moveable* areas and regions that belong to *non moveable* areas. The method relies on interactive perception. It includes a classifier, called collaborative mixture models (CMMs) presented in the previous chapter (Chapter 4) that is trained online and a query strategy that selects the regions upon which to focus. The query strategy aims to reduce uncertainty in the classification and balance the number of samples in each class. A change detector determines the class of the region with which the robot has interacted. The corresponding data are added to a dataset used to train the classifier. The approach generates a relevance map segmenting objects from the background. This information can be used to bootstrap an object discovery method, reducing the assumptions on the structure of the environment and thus paving the way to approaches that can adapt to a wider range of environments. The approach has been tested on setups of increasing complexity using simulations and a real PR2 robot.

In the next chapter, each component of the CMMs classifier is evaluated to assess their utility. These evaluations will draw leads to enhance the training algorithm CMMs. As the outcome of the next chapter, CMMs is modified for better performances.

Chapter 6

Exstensive study of CMMs

Contents

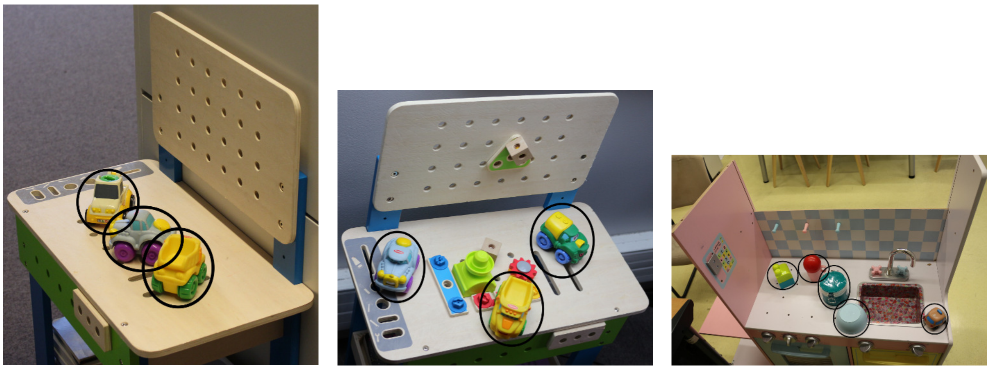
6.1 Introduction

In this chapter, experiments are conducted to study the different parts of CMMs and identify their utility and properties. The experimental setups are constituted of an environment (see figure 6.1) and a Kinect v2 RGB-D camera in front of it. Thus, the following experiments are made on a real video stream. The experiments follow the 5 steps described in section 5.4 but instead having a robot interacting with the environment, an expert is used as labeling system. The expert is built as described in section 5.5.1.

The goal of an experiment is to train a classifier to produce a segmentation of objects (circled in black in figure 6.1) and the background (a relevance map). In a "regular" experiment with a robot (like for the *real world experiment* described in section 5.5.1) these objects would be considered as "moveable".

These experiments aim at testing the learning algorithm with all its different configurations with a setup close from experiments with a real robot. These experiments are quicker to execute as there is no robot, and thus permit a lot of experiments. All the following results are computed with 10 replications per experiment.

Precision, recall, and accuracy scores presented in the next section are computed by following the formula described in section 5.5.2 and are computed during an experiment like in the previous chapter.



(a) Simple workbench setup (b) Second workbench setup (c) Kitchen setup

Figure 6.1: Pictures of the environments used for the experiments of this chapter. The objects being considered as "moveable" are marked by a black circle.

6.2 Splitting and Merging

6.2.1 Protocol

MNIST dataset In this section, the first series of experiments are made on the MNIST dataset (handwritten digits made by LeCun et al. [2010]). The classifier is trained on this dataset in batch mode (see section 4.5). These experiments are conducted to test the role of splitting and merging operations by varying α parameter and by using or not the loglikelihood as a criterion for splitting or merging. Indeed, α controls the sensibility of splitting and merging: when α is equal to 1 there is no splitting or merging. Each experiment consists of 10 replications of training of CMMs on MNIST with α varying between 0 and 1 with a step of 0.1. A training is 300 epoch, at each epoch 100 samples are added to the training dataset.

Also, these experiments permit to test CMMs on a 10 classes problem. In the thesis, all the other experiments involve 2 classes.

On real environment The split and merge mechanisms are tested by varying α parameter and by setting different maximum numbers of components by mixture. In the previous chapter (5), there was no maximum number of components, thus the number of components was increasing proportionally to the number of samples gathered and a decrease of classification performance was observed. Setting a maximum number of components could stabilize the classification quality.

6.2.2 Results

In the previous chapter (5), split and merge operations were validated by comparing the loglikelihood of the models before and after a merge or a split. Figure 6.2 shows the generalization error of a classifier trained on MNIST dataset with and without using the loglikelihood. Using loglikelihood as a value to optimize does not lead to the best results. While without the use of loglikelihood, the classification error decreases quickly and converges to a value around 0.2, with the use of loglikelihood the error converges quickly to a value close to 0.4 and seems to be less stable. Moreover, for these experiments, only 40 epochs for each replication have been conducted as computing the loglikelihood of the model at each iteration is very costly and can multiply by 10 the computation time. These experiments have been conducted with α equal to 0.8.

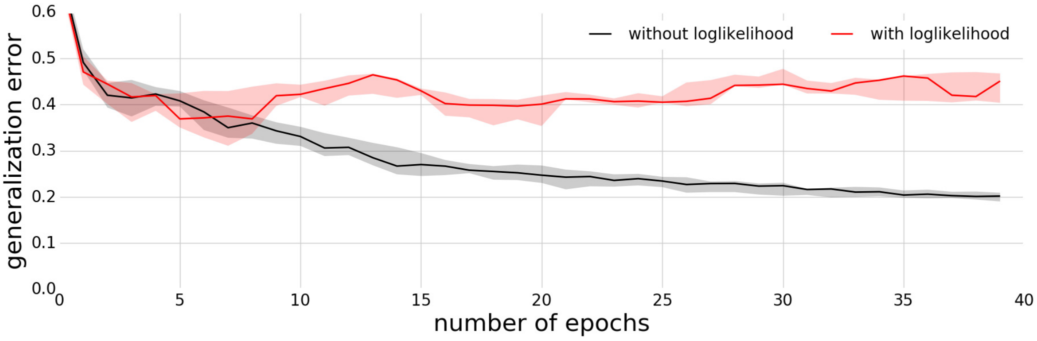


Figure 6.2: Classification error on the test dataset over the epoch during a training on MNIST dataset in batch mode with loglikelihood minimization in red and without in black.

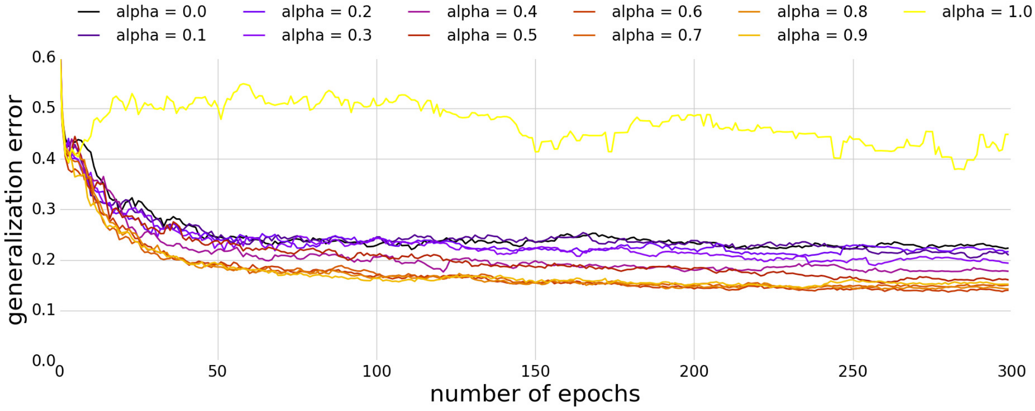
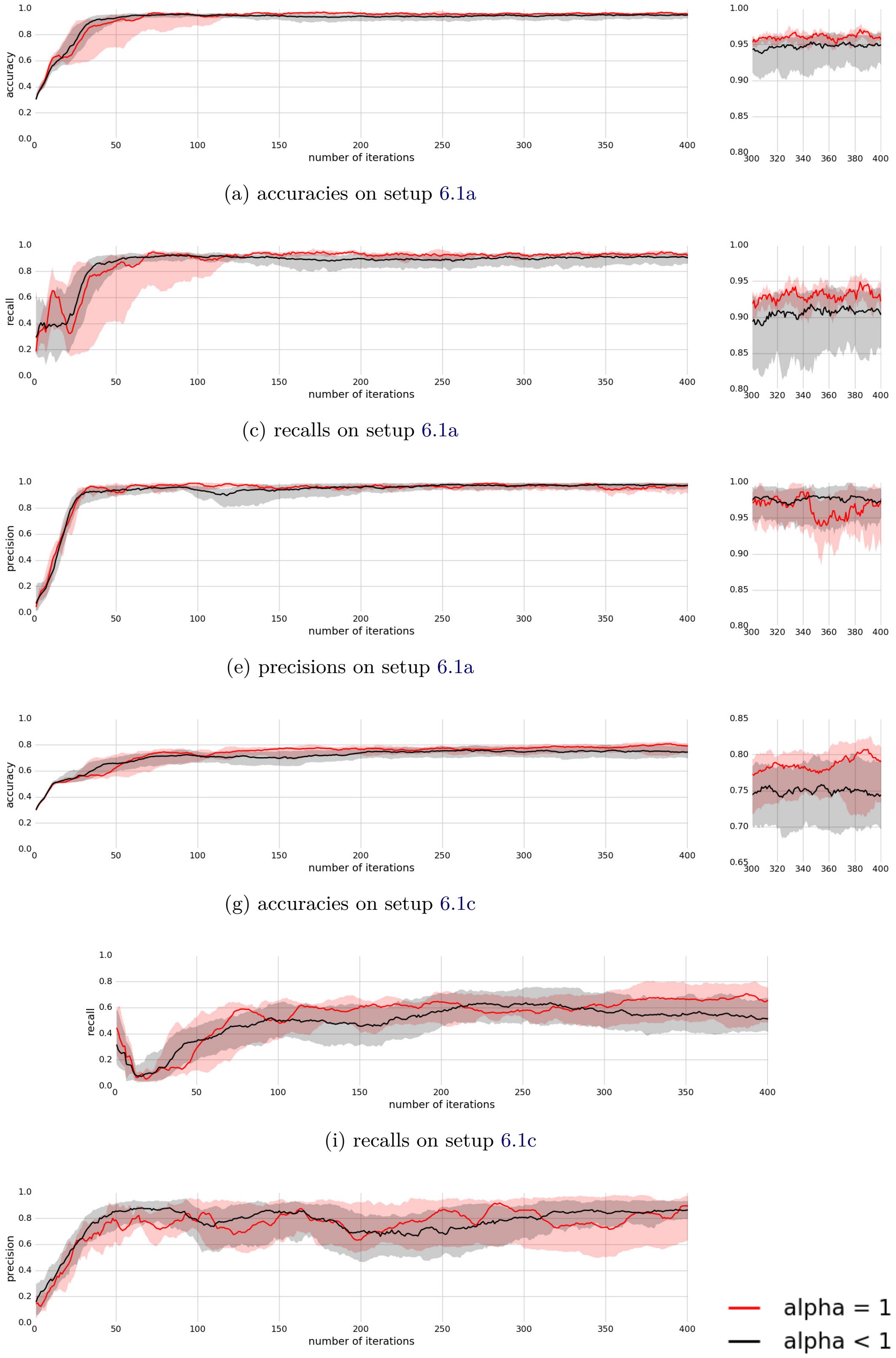


Figure 6.3: Training on MNIST dataset in batch mode with parameter α varying between 0 and 1 with a step of 0.1. For each value of α, 10 replications have been conducted. The curves correspond to the median of the replications.

Figure 6.3, shows the classification error on MNIST dataset through 300 epochs of training. The parameter α varies between 0 and 1. First, the highest error is reached by the trained classifier with α equal to 1, (it has a minimum error of approximately 0.38). The classification seems to be also very unstable through the epochs compared to the other training sessions. For the other values of α, the errors vary between 0.23 and 0.15 after convergence. The best performing classifiers are those with alpha between 0.6 and 0.9. These results show that, for the MNIST dataset, split and merge operations are useful since α equal to 1 mean no split nor merge. Moreover, CMMs reach a quiet good result on a 10 classes problem. The MNIST dataset has 60000 pictures in training dataset and in these experiments, only 30000 examples were processed by CMMs. These experiments have been conducted without the loglikelihood.

Figures 6.4 represents precision, recall, and accuracy over the iterations during training sessions on the simple workbench (6.1a) and the kitchen setups (6.1c) with α varying between 0 and 1. Unlike training conducted on MNIST dataset a limit of 4 components per mixture model has been set. In these figures and as they show a limited variability, all the replications with α less than one are gathered in one curve for better clarity. So, the red curve includes 10 replications, while the black curve includes 100 replications. There is not a large difference between the median of training sessions with α equal to 1, i.e. without splitting and merging, and training sessions with α less than 1, unlike for training on MNIST dataset. For the simple workbench setup (6.1a), training sessions with α equal to 1 generate better results than with α less than 1. This is expected. As this setup is very simple, the classes are most likely linearly separable, thus, one component per mixture is enough. But, with α less than 1, the classifier also reaches a significant performance, even slightly better than α equal to 1 on precision. For the kitchen setup (6.1c) the performance is lower, as expected since this environment is more complex. The variability of runs with α equal to 1 is higher than with α less than 1 which is clearly visible on plots 6.4i and 6.4j. On this setup, accuracies and recalls, for alpha less than 1, are below those for alpha equal to 1 while the median of precisions is unstable with high variability for alpha equal to 1 but stable with a low variability for alpha less than 1.

These results suggest that without splitting and merging the classifier will reach a better recall with a lower precision when with split and merge the classifier will reach a better precision with a lower recall. This conclusion is coherent with the results presented in figure 5.12 of section 5.6. In other words, with splitting and merging the relevant areas may be too small while without splitting and merging the relevant areas may be too large.



(a) accuracies on setup 6.1a

(c) recalls on setup 6.1a

(e) precisions on setup 6.1a

(g) accuracies on setup 6.1c

(i) recalls on setup 6.1c

(j) precisions on setup 6.1c

Figure 6.4: Accuracy, precision, and recall for the simple workbench (6.1a) and for the kitchen (6.1c) with α varying between 0 and 1 with a step of 0.1. For the sake of clarity, all replications for α strictly less than 1 are gathered in the black curve. For each value of α, 10 replications have been conducted.

The maximum number of components per mixture is a new hyperparameter introduced in this chapter which should be specific to the complexity of the problem, in particular to the (non-)convexity or (non-)linear separability of the classes (see chapter 4). Experiments have thus been conducted on the second workbench and the kitchen setups (6.1b, 6.1c) with different maximum number of components and with no limit of numbers of components to assess the influence of this parameter.

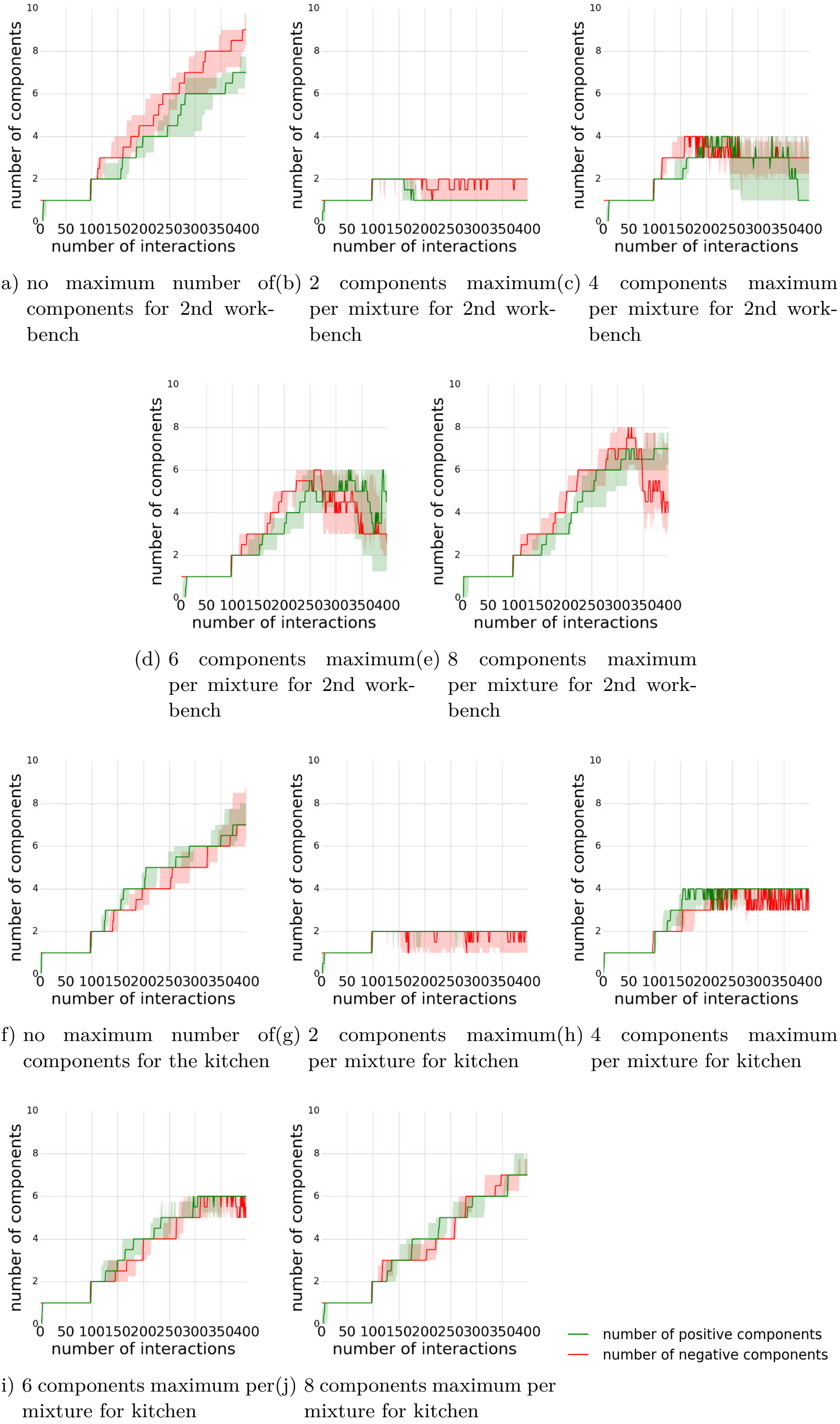
Results are presented in figures 6.5, which plot the number of components over the iterations, and in figures 6.6 and 6.7 which present precision, recall, and accuracy scores.

On the second workbench environment (6.1b) with a maximum number of components per class the model reaches the maximum and then decreases, while without a maximum, the number of components keeps increasing. The maximum number of components is reached after a number of iterations depending on the maximum number of components: for 2 the maximum is reached after 100 iterations, for 4 after around 200 iterations, for 6 around 250 and for 8 around 300. This is expected because of the theoretical limit introduced by the intersection condition (see section 4.6.1). The most stable and best performing runs seem to be with 4 maximum components per class as shown on 6.5c. While with 2, 8 and no limit the performances decrease after a certain iteration. With 6 maximum components the best performance is reached at the end, thus exploration would need more iterations.

On the kitchen environment (6.1c), the number of components reaches its maximum and then stagnate for 2, 4, and 6 maximum number of components. For 8 maximum components the stagnation is reached after 400 iterations, thus, for this experiment, the budget was increased to 800 iterations. The results are the best with 8 components, but the number of components does not decrease unlike with the second workbench setup (6.1b). Probably, for this environment, the model needs more than 8 components per class to converge. As expected, with no limit the number of iterations keeps increasing.

The maximum number of components seems specific to the environment and will be difficult to fix a priori apart from the budget also fixed. This parameter seems to be tightly linked to the budget, thus, the best way to choose a value for the maximum number of components is to fix it in relation to the budget. The perfect solution would be to have an ending criterion which would remove the need to fix a budget.

All these experiments have been conducted without loglikelihood and an α fixed to 0.6.



(a) no maximum number of components for 2nd workbench

(b) 2 components maximum per mixture for 2nd workbench

(c) 4 components maximum per mixture for 2nd workbench

(d) 6 components maximum per mixture for 2nd workbench

(e) 8 components maximum per mixture for 2nd workbench

(f) no maximum number of components for the kitchen

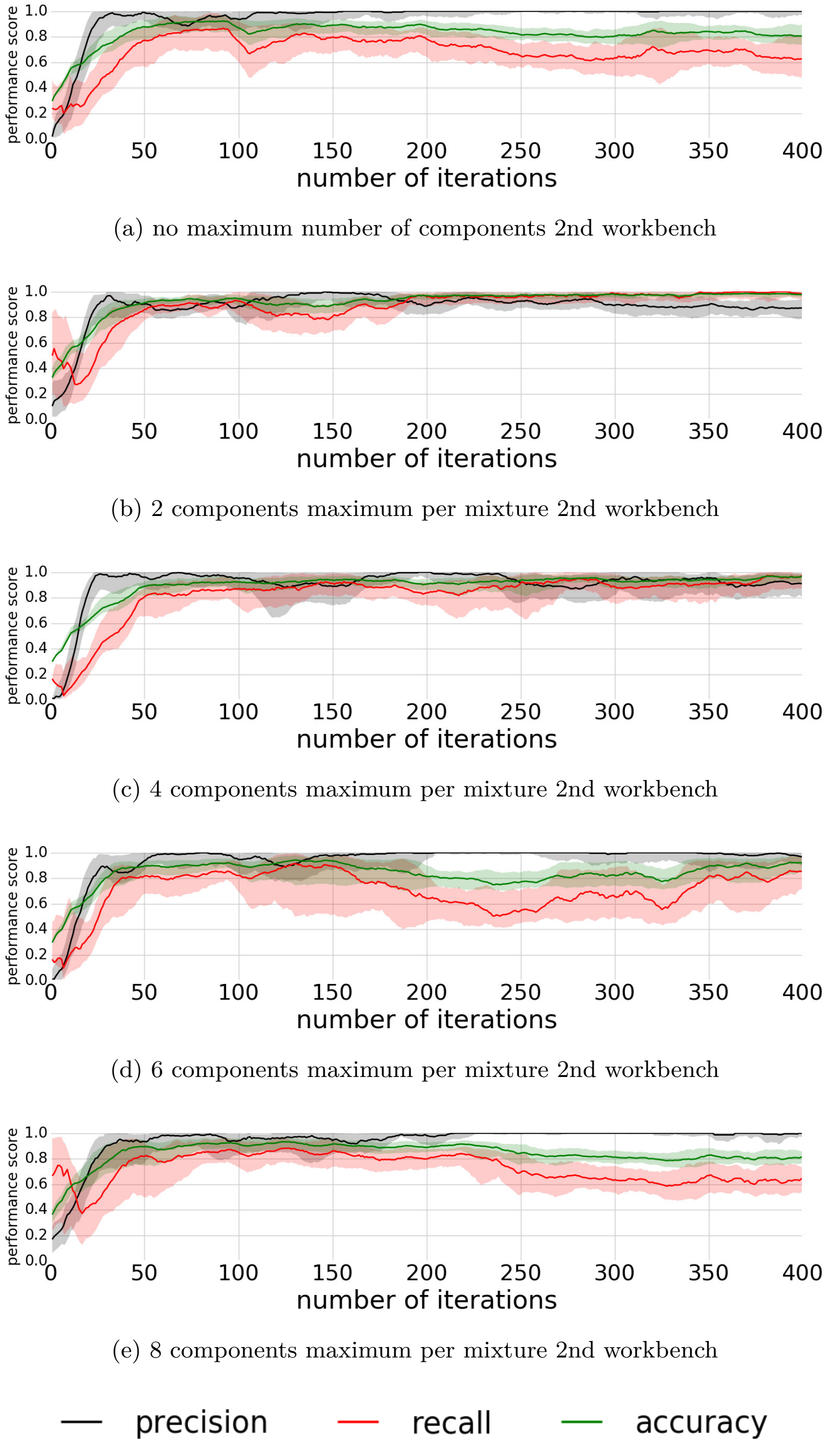
(g) 2 components maximum per mixture for kitchen

(h) 4 components maximum per mixture for kitchen

(i) 6 components maximum per mixture for kitchen

(j) 8 components maximum per mixture for kitchen

Figure 6.5: Number of components for different maximum number of components for experiment conducted on the second workbench and the kitchen setups (6.1b,6.1c)



(a) no maximum number of components 2nd workbench

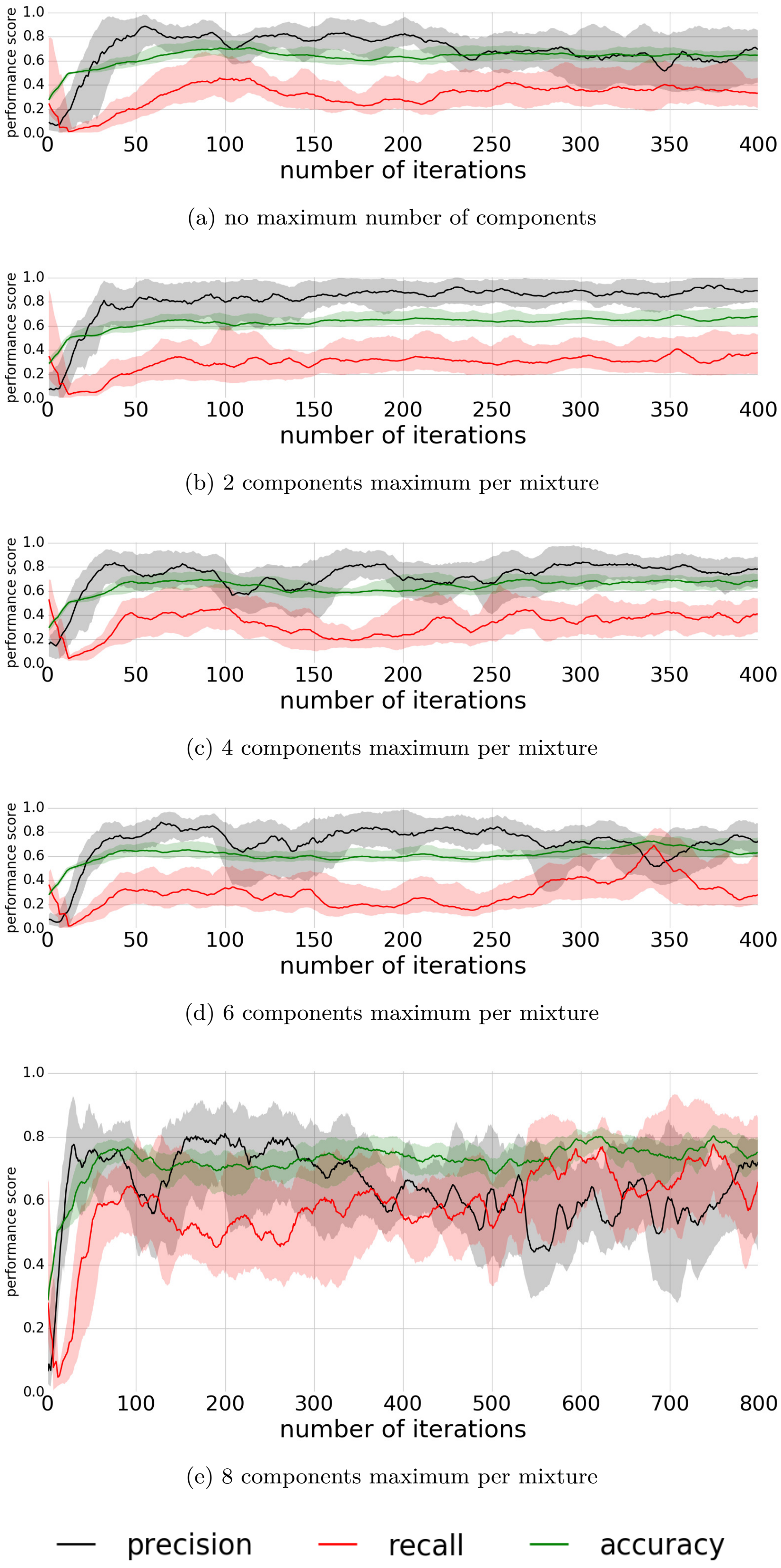
(b) 2 components maximum per mixture 2nd workbench

(c) 4 components maximum per mixture 2nd workbench

(d) 6 components maximum per mixture 2nd workbench

(e) 8 components maximum per mixture 2nd workbench

Figure 6.6: Precision, recall, and accuracy for different maximum number of components for experiment conducted on the second workbench setup (6.1b)



(a) no maximum number of components

(b) 2 components maximum per mixture

(c) 4 components maximum per mixture

(d) 6 components maximum per mixture

(e) 8 components maximum per mixture

Figure 6.7: Precision, recall, and accuracy for different maximum number of components for experiment conducted on the kitchen setup (6.1c)

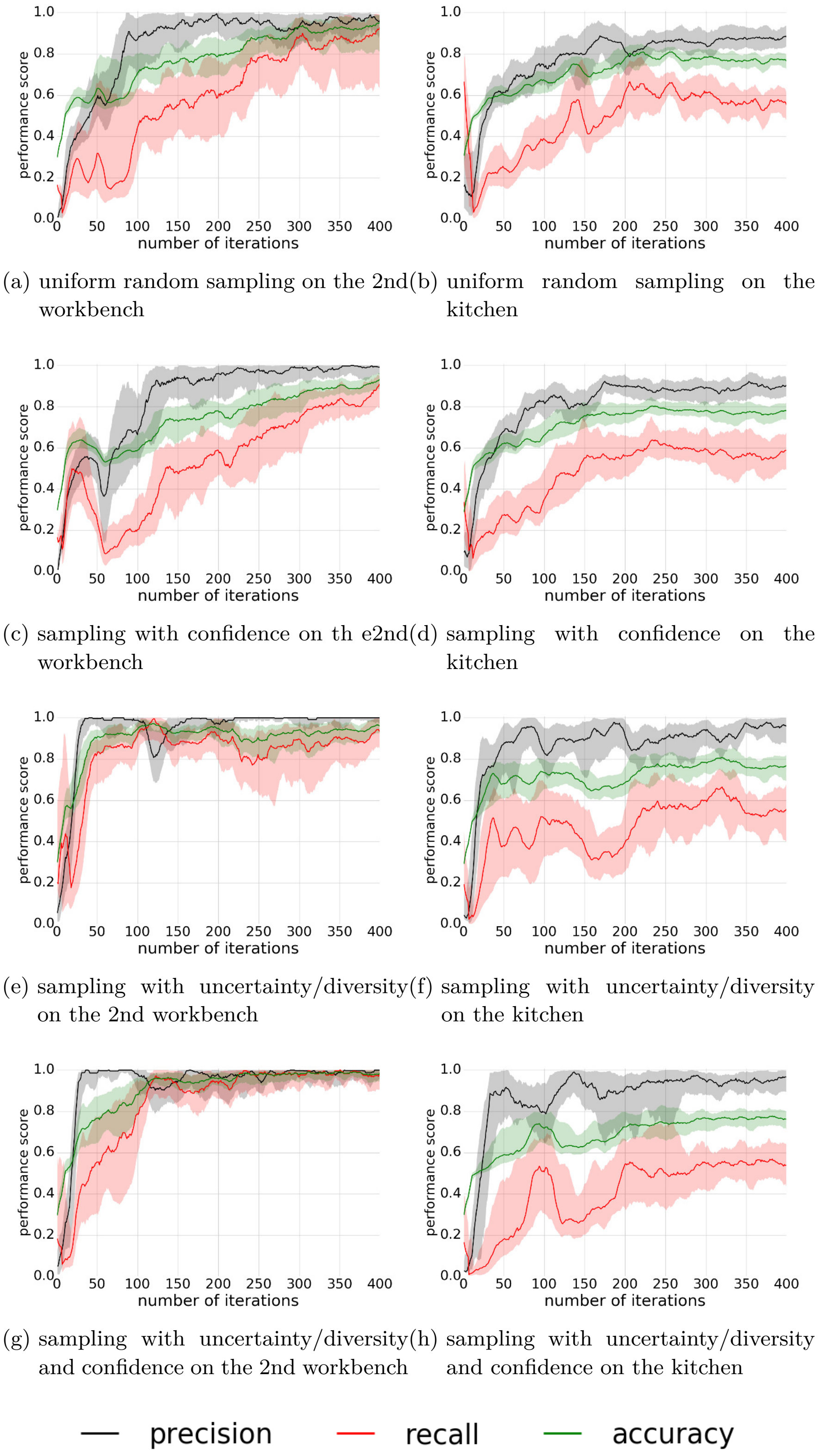
6.3 Query strategy

This section presents an evaluation and comparison of the different methods introduced in section 4.6.2. The protocol for the following experiments is the same as for the previous sections. The experiments are conducted with α equal to 0.6, without loglikelihood and a maximum limit of 4 components. The tested query strategies are the following: uniformly random, uncertainty/diversity alone, confidence alone, and uncertainty/diversity with confidence combined. Figures 6.8 show the plots of precision, accuracy, and recall scores trained with these different strategies. Training was conducted on the second workbench and kitchen environments (6.1b, 6.1c).

On both environments, with random sampling the training converges, but much slowly than with uncertainty/diversity sampling (see 6.8a, 6.8e, 6.8b, and 6.8f). Indeed, with uncertainty/diversity sampling the accuracy, recall, and precision scores quickly reach a plateau and then increase slowly or oscillate.

Quick convergence is an important feature for learning relevance map as it is online and the query strategy relies on the classifier prediction. Thus, to have an efficient query strategy the classifier needs a relatively meaningful prediction quickly. Moreover, the robot could fail its action and collects false negative samples and as shown in chapter 5 on figures 5.9 the number of positive samples gathered with a uniform random sampling is very low. Thus, with random sampling the classifier will converge slowly and may not converge within the budget because too few numbers of true positive samples would be gathered.

Confidence seems to have little influence when comparing random sampling and confidence sampling (see figures 6.8a, 6.8c, 6.8b, and 6.8d). But according to figures 6.8e and 6.8g confidence sampling enhances the training. Thus, uncertainty/diversity sampling allows training to converge very quickly and confidence sampling in certain cases increases the quality of the classification.



(a) uniform random sampling on the 2nd workbench

(b) uniform random sampling on the kitchen

(c) sampling with confidence on th e2nd workbench

(d) sampling with confidence on the kitchen

(e) sampling with uncertainty/diversity on the 2nd workbench

(f) sampling with uncertainty/diversity on the kitchen

(g) sampling with uncertainty/diversity and confidence on the 2nd workbench

(h) sampling with uncertainty/diversity and confidence on the kitchen

Figure 6.8: Precision, recall and accuracy for experiments conducted on the 2nd kitchen setup (6.1b). The experiments were conducted with different query strategies.

6.4 Supervoxel features

6.4.1 Protocol

In the following section, 6 different features are compared :

- *Mean color and mean normal* of a supervoxel. The color is in the RGB domain. This is a naive feature space with 6 dimensions.

- *Color histogram* of a supervoxel extracted on CIELab color encoding. One histogram of 5 bins is computed separately on each dimension ($L * a * b$) and then they are concatenated. This feature space has 15 dimensions.

- *Mean FPFH* of a supervoxel. FPFH is extracted on a pointcloud including the targeted supervoxel and its neighbors. One FPFH for each point is obtained, then the average of these histograms is computed to obtain the final feature. This feature has 33 dimensions.

- *Central FPFH* of a supervoxel. FPFH is extracted on the central point of the pointcloud including the targeted supervoxel and its neighbors. The radius of neighborhood to compute FPFH is set to the size of a supervoxel, thus the central point FPFH takes into account the whole considered pointcloud. The central point is the centroid of the targeted supervoxel. Like mean FPFH, this feature has 33 dimensions.

- The concatenation of *color histogram and mean FPFH* which has 48 dimensions. This is the feature used in chapter 5 to train the CMMs classifier and to compute the relevance map.

- The concatenation of *color histogram and central FPFH* which also has 48 dimensions. This is the feature used for the experiments of the other sections in this chapter.

For this section, the experiments are conducted on the second workbench setup (6.1b). A first experiment consists in training a classifier with samples of the concatenated color histogram and central FPFH feature. At each iteration, a sample is extracted for each of the above-mentioned features. In this experiment, the choice distribution map is computed thanks to the classifier trained on the concatenated central FPFH and color histogram feature. Then, 4 other experiments are conducted by training a classifier based on color histograms, central FPFH, central FPFH combined with color histogram and mean FPFH combined with color histogram. All the experiments are conducted with α equal to 0.6, without loglikelihood and a maximum limit of 4 components.

To compare these features, a separability score is computed with a K-means clustering on the datasets collected during the experiments. The data are clustered without their labels. Several K-means are applied with a number of clusters from 2 to 10. The separability score is computed according to the equation 6.1 :

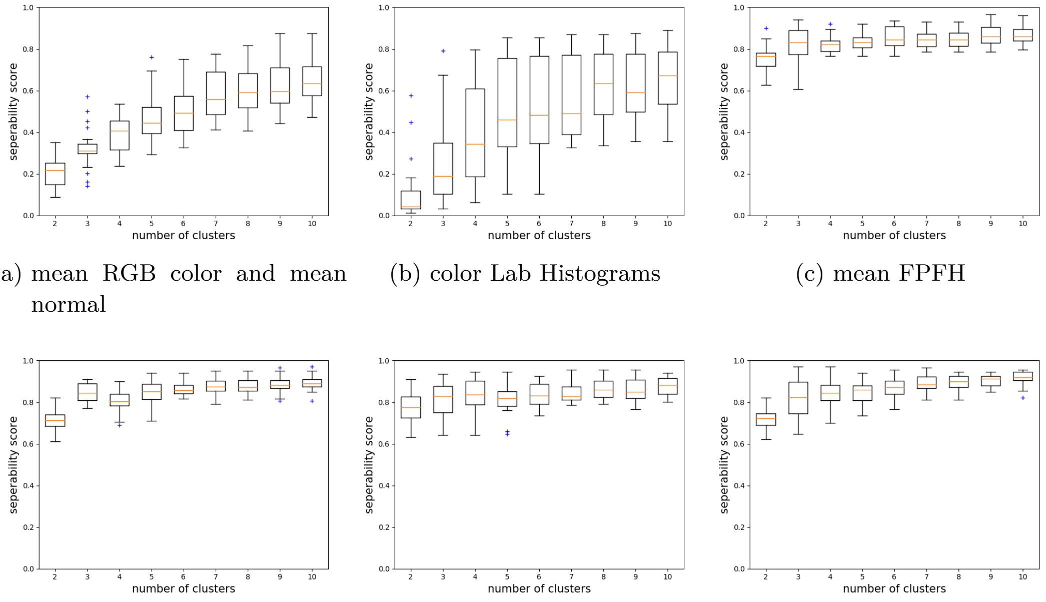
$$separability = \frac{1}{|S|}\sum_{i=0}^{K}\sum_{j=0}^{|c_i|}|\delta_{l_{ij}=1} - \delta_{l_{ij}=0}| \tag{6.1}$$

Where S is the dataset (see 4.5), K the number of clusters of the K-means algorithms, c_i is the ith cluster, $\delta_{x=y}$ is the Kronecker symbol equal to one when the proposition $x = y$ is true, and l_{ij} is the label of the ijth data.

To compute this score, the number of samples from the same class is counted in each cluster. Then, the number of samples from class 0 and 1 is subtracted. Thus, the more a cluster contains samples from only one class, the higher its score. The separability score is equal to 1, if each cluster contains only one class, and is equal to 0 if each cluster contains the same number of samples of each class. This score is consistent only if the dataset contains the same quantity of samples from each class. For instance, for a K-means with 2 clusters with a dataset containing 100 samples of both classes, separability score is equal to 1 if each cluster contains 100 samples of only one class. The separability score is equal to 0 if each cluster contains 50 samples of each class.

6.4.2 Results

Results are plotted in figures 6.9 and 6.10.



(a) mean RGB color and mean normal

(b) color Lab Histograms

(c) mean FPFH

(d) central FPFH

(e) mean FPFH and color Lab Histograms

(f) central FPFH and color Lab Histograms

Figure 6.9: Box plots of separability scores on dataset collected in experiment conducted on the second workbench setup 6.1b

For all features, the separability score increases with the number of clusters.

This is expected as more clusters are allowed, easier the separation of two classes data because if the number of clusters is equal to the number of samples the separability score is obviously equal to 1. So, the separability score grows like a sigmoid as the number of clusters grows. The mean color and mean normal features have low separability with the best score at 0.6 for 10 clusters. This is expected as mean values attenuate the differences. FPFH based features have high separability scores even with 2 clusters. The color histogram shows a high variability on separability. There is no significant difference between mean and central FPFH features. Geometrical features are more discriminative than color features which is expected as FPFH has a higher dimensionality and is more informative in relation to the problem. High dimensional and non redundant features are expected to be more discriminative. For instance, convolutional neural networks build features vector with a thousand values which is one of the reasons of their efficiency. Moreover, FPFH is designed to be highly discriminative on geometry which is intuitively the best information to consider to solve the classification problem addressed in this chapter.

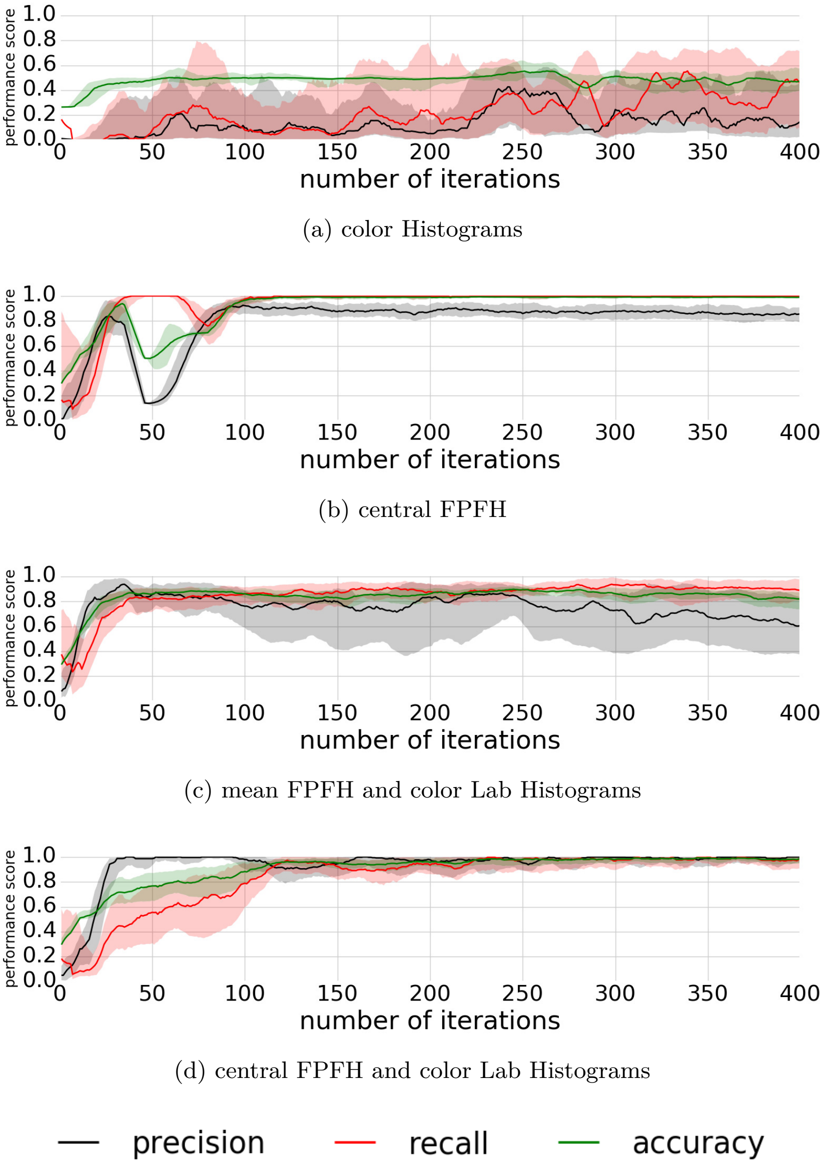
Classification on color histogram alone performed poorly as shown in figure 6.10a which is expected regarding the separability score of this feature. Classification on central FPFH alone has a high performance score (see figure 6.10b). However, its recall score decreases slowly. Regarding the separability scores, central FPFH or mean FPFH should be chosen alone without the color. However, classification on central FPFH concatenated with color histogram has a higher performance than central FPFH alone (see figure 6.10d). While there is a higher variability over the replications than with central FPFH alone, the precision, recall, and accuracy converge more quickly and the recall does not decrease. Finally, classification based on mean FPFH concatenated with color histogram the recall and the accuracy decrease after convergence. The variability of these scores is also high.

Therefore, according to these results, the best feature to choose is the central FPFH concatenated with color histograms.

6.5 Discussion and Future works

Modification of the CMMs training algorithm According to the results presented in the previous section, some modifications of the CMMs training algorithm need to be done. The loglikelihood as a condition for splitting or merging components is not leading to a sufficient classification performance, thus, it should be removed from the algorithms 2 (split operation) and 3 (merge operation). It lets the intersection criterion as the only condition for splitting or merging. This modification simplifies significantly these algorithms as no candidate models need to be computed anymore. The second modification is the addition of a maximum number of components to the split algorithm 2. The modified algorithms are written in 7.3 in the next chapter.

The maximum number of components is a parameter that is specific to the envi-



(a) color Histograms

(b) central FPFH

(c) mean FPFH and color Lab Histograms

(d) central FPFH and color Lab Histograms

Figure 6.10: Precision, recall, and accuracy of experiments conducted on the second workbench setup 6.1b with different features.

ronment. Thus, setting a value of this parameter that is common to all environment is an issue. The intersection condition defined in equation 4.10 introduce a theoretical minimum size to apply merge or split operations which is the dimensionality δ of the feature space. Here, the features space is $\delta = 48$ dimensional which defines a minimum size of 48 samples for a component to be split or merged. As a babbling has a fix budget of collected samples I, we can expect a theoretical maximum number of components $K_{max} = \frac{I}{\delta}$. As shown in the result of section 6.2, without a limit, the number of components exceeds K_{max} (see 6.5a and 6.5f). Consequently, in the following experiments, this parameter is set according to the budget and the dimensionality of the features space. The budget is a critical hyperparameter because the more complex an environment is, the more samples are needed to reach a sufficient classification. This also corresponds to the number of components which should increase with the complexity of the environment. Finally, the mean FPFH feature is replaced by the central FPFH feature which is more discriminative and lead to better performance.

Ending criteria or convergence problem The results from this chapter and from the previous chapter show that convergence is not always certain and even after convergence, performance could decrease. A criterion to end the exploration needs to be defined. This question is hard as the exploration is autonomous with no external information about the environment. The criterion must assess the quality of the classification and take into account the discovery of novel elements, to deal with open environments. Moreover mislabeled samples could be collected due to action or effect detection failures, as shown in chapter 5. If the effect detector can be engineered to fail a minimum number of times, the number of failed interactions is hard to predict because of the dependence on the environment structure. Indeed, the more complex the environment, the more interaction failures can occur.

To solve this issue different solutions are possible. A subset of the collected samples can be kept as a test set to compute the generalization error. However, a test set must be sufficiently different from the training dataset. This could be problematic in an online learning algorithm. Another solution could be to reinitialize the classifier with regular batch sessions after a certain number of collected samples. This could reduce the dependency of the classifier quality to the order of arrival of the samples. To handle the problem of open environments or mislabelled samples, a mechanism of unlearning could be added like in the online random forest algorithm of Saffari et al. [2009]. This could be implemented by weighting the samples to assess their quality. These weights could be computed on the basis of a statistical measure.

To track the classification quality, unsupervised measures can be used like the loglikelihood or the variance of the model. The loglikelihood could be used as it is linked to the expectation of the model and to the error of classification. The variance is interesting because of its link with the expected error of prediction. This relation was demonstrated by Geman et al. [1992] with, as result, the equation 6.2:

$$E_T[(\tilde{l}-l)^2|X] = E[(l-E[l|X])^2] + (E_S[\tilde{l}] - E[l|X])^2 + E_S[(\tilde{l}-E_S[\tilde{l}])^2] \tag{6.2}$$

Where $E_S[.]$ is an expectation over the dataset S collected during an exploration, $E[.]$ is an expectation over the conditional density $P(l|X)$,$E_T[.]$ is an expectation over both, $\tilde{l}$ is the label predicted by the classifier and l the true label of the input X.

This proposition could be useful, as the left part of the equation is the expected error, on the right part, there is first the model independent *noise* which includes all the variability due to the labelling system, that is here made up with the effect detector and the action primitive. Then, there is a *bias* term which represents the model error in relation to the density $P(l|X)$ and finally the variance of the model. Thus, this relation between the variance and the expected error carries information which could be used as an end criterion. This being said, these values require intensive computations and an approximation of the expectation of the unknown probability density $P(l|X)$ must be computed.

Another interesting measurement is the Kullback-Leibler divergence defined in equation 6.3 (Kullback and Leibler [1951]). This measure is interpreted in machine learning as the *information gain* of Q in comparison to P. It is linked to the Shannon entropy and also to the loglikelihood.

$$D_{KL} = -\sum_{x \in X} P(x) log(\frac{Q(x)}{P(x)}) \tag{6.3}$$

Where, X is the input space, Q and P two distributions.

Equation 6.3 can be used as an objective function by comparing the model before and after an update. Also, an EM algorithm can be expressed as two maximizing steps (Neal and Hinton [1998]) in which the objective functions contain the Kullback-Leibler divergence.

Also, the mean integrated squared error could be used, as it is a distance between the estimated distribution and the unknown target distribution. Kristan et al. [2008] proposed a recursive formula to approximate this measure, but it needs an approximation of the unknown distribution which makes its computation more complex.

Finally, a threshold on the entropy or uncertainty is an option. Indeed, the exploration seeks to reduce the uncertainty or the entropy of the model, thus, considering a minimum entropy or uncertainty as sufficient could be an end criterion. That would imply to compute a global uncertainty measurement and the entropy is heavy to compute.

The resilience of the classifier to mislabelled has not been tested in this chapter.

6.6 Conclusion

The main components of CMMs have been tested in this chapter. First, the input of the split and merge features have been validated by testing a large range of α values. This hyperparameter controls the split and merge sensitivity. Moreover, the variability of the results when α is changed is relatively low. The contribution of the uncertainty/diversity and the confidence as defined in section 4.6.2 to generate the choice distribution map, has been demonstrated. The contribution of confidence seems to be weak but sufficient to be kept. Finally, a comparison between different features leads us to choose the concatenated central FPFH and color histogram feature (as defined in section 6.4) for the experiments conducted in the next chapter.

The results also show that the maximization of the loglikelihood as a condition for splitting and merging should be removed and that a maximum number of components has to be fixed in the split operation.

The CMMs algorithm has been evaluated on the MNIST dataset and on three real environments. On MNIST dataset, which contains 10 classes, the classification quality reaches 0,86 after having processed half of the dataset (30000 from the 60000 provided). in the real environment, the accuracy reaches a value of almost 1 on the simplest and between 0.7 and 0.8 for the hardest environments. CMMs demonstrates then a sufficient performance to reach its goal.

Chapter 7

Affordances Map

The results and text of this chapter have been partially published in the following articles :

> Le Goff, L. K., Yaakoubi, O., Coninx, A., and Doncieux, S. (2019). Building an Affordances Map with Interactive Perception. arXiv preprint arXiv:1903.04413.

Other contributors:

- Stéphane Doncieux, Sorbonne Université (Thesis supervisor)
- Oussama Yaakoubi, formerly, in 2018, Sorbonne Université (Engineer)
- Alexandre Coninx, Sorbonne Université (Maitre de conférence)

Contents

7.1 Introduction

This chapter presents an extension of the work introduced in chapter 5. In this previous chapter, a relevance map is built based on data collected thanks to the interaction of a robot with an environment through a push primitive. This approach is within the scope of interactive perception as it learns a representation of the world through interactions with an environment. This is directly linked to the concept of affordances introduced by J.J. Gibson [1966, 1979]. An affordance is a relational property of the agent-environment system. According to Gibson, an agent perceives the world through the actions afforded by the elements in it. An approach based on interactive perception allows a robotics system to learn the link between visual features and the actions used, in other words, features that afford an action.

Without changing the core and the structure of the framework, relevance maps relative to other primitives could be learned. In this chapter, we combine experiments made with a push primitive, with a lift primitive, and with a push button primitive. These maps are then combined to produce an affordances map. The problematic of the following study is: *How a robot with a toolbox of primitives can build a perception based on affordances by autonomously exploring its environment ?* This affordances map is a starting point for more complex tasks or actions.

This chapter is organized as follow : A description of the foundations and of the definition of the affordance concept is given in section 7.2.1, then in section 7.2.2, formalizations of affordance in the context of robotics are described and in section 7.2.3 several works on affordance learning based on local features are reviewed. The proposed method to learn affordances is explained in section 7.3 and the experimental protocol used to evaluate the method is presented in section 7.4. Finally, results are presented in section 7.5.

7.2 Affordances

7.2.1 Foundation and Definition(s)

The concept of affordance was first introduced by J. J. Gibson in 1966 :

> *"When the constant properties of constant objects are perceived (the shape, size, color, texture, composition, motion, animation, and position relative to other objects), the observer can go on to detect their affordances. I have coined this word as a substitute for values, a term which carries an old burden of philosophical meaning. I mean simply what things furnish, for good or ill. What they afford the observer, after all, depends on their properties."* Gibson [1966]

He refines his definition in a later book written in 1979 :

> *"The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. The verb to afford is found in the dictionary, but the noun affordance is not. I have made it up. I mean by it something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment."* Gibson [1979]

With this concept, Gibson wanted to highlight that objects have inherent "values" and "meanings" which could be perceived by an agent and could be linked to its possible actions on those objects. And thus, an animal or a human perceives the environment through the actions he can use according to his abilities and the elements in the environment. But as the two quotes show, the definition of his concept of affordances is vague. So, ecological psychologists work to state a more precise definition. Some core aspects have been discussed during the second half of the XXth century. Three main issues were discussed :

- Is an affordance a property of the environment or an emergent relationship in the agent-environment system ?
- Is the perception of affordances direct or does the agent need to build an internal representation of affordances ?
- Do affordances exist independently from the perception of an agent ?

Two opposite main views are proposed by psychologists. In the first one, affordances are properties of the environment which exist always and are directly perceived by the agent. In the second one, affordances are emergent relational properties of the agent-environment system, thus affordances do exist only in the agent-environment system. In this view, the direct perception of the affordances is not always assumed.

Turvey [1992], a psychologist, defines affordances as *dispositional properties* of the environment. Those properties are visible for the agent only under certain conditions: when the agent can apply an action on the focused object. For instance, a stone affords grasping and throwing only if the agent is able to grasp it. So, according to Turvey's view, affordances exist independently from an agent perception.

Experiments conducted in ecological psychology (Warren [1984], Warren Jr and Whang [1987], Mark [1987]) show that the ratio between agents body dimensions and the dimensions of the environment is important in deciding which action is possible or not. This ratio is a feature that the agent perceives and uses. This fact tends to define affordances as a relationship between the agent and its environment.

Chemero [2003] and Stoffregen [2003] define affordance as a relational property of the agent-environment. In their view, the agent can directly pick up information that already exist in the agent-environment system. In the case of Turvey, the agent would have to do further processing to build such information. In Chemero's view, even if affordances exist without the presence of an agent, to observe affordances the potential presence of an agent is needed. Thus, without an agent, affordances cannot be studied.

Vera and Simon [1993] consider affordance as an internal symbolic representation which is learned from a semantic mapping between a symbolic perception and an action representation. This implies that affordances are not directly perceived and they do not always exist but they are rather learned. In a similar view, Steedman [2002a,b] assumes that an agent needs to learn about the function of objects to perceives the affordances. Since the agent has the knowledge of the object *functionality*, it will directly perceive the associated affordance. Steedman in his computational model does not consider the perception part but only a symbolic formalization of affordances. He considers affordances as a relation between objects, actions, and events.

Norman, in his book Psychology of Everyday Things (Norman [2013]), a book about design, defines affordance as a relationship and adds that affordances are not always visible. The job of a designer is to "make things visible". Our everyday environment is designed to be easy to analyze, to perceive the affordances thanks to *signifiers*. A *signifier* is a concept introduced by D. Norman. He defines it as something visible which show us the affordance. So, our homes, workplaces, tools, electronic devices, and so on, are designed to be discoverable. The *discoverability* characterizes how the functionality of an object is understandable without external help.

From this literature review about affordances, several conclusions come up :

- Affordances *emerge* from the relation between the agent and the environment;
- *Functionality* is an inherent property of objects or parts of the environment. A functionality could become an affordance if the agent has some knowledge about it and if the agent is able to use it;
- Affordances are not always easy to "see". Therefore learning and exploration could be needed to perceive affordances. *Signifiers* could be built to help an agent to perceive affordances.

In this thesis, we state that an affordance is an emergent relationship in the agent-environment system. Thus, an affordance is a relationship between a sensory signal, the agent skills and the possible effect that would result from the agent's actions. Affordances are learned from experiences of the agent interacting with the real world. When learning is done, affordances are directly perceived. Moreover, for the affordances to be learned, the environment needs to have distinctive and coherent sensory signals associated with actions and effects, in other words, they need to be discoverable.

7.2.2 Affordances in Robotic

Affordances have raised a lot of interest in the developmental robotic community these last ten years. The multiple surveys written by robotics researchers on affordances show this interest : Sahin et al. [2007], Horton et al. [2012], Min et al. [2016], Jamone et al. [2016], Zech et al. [2017].

Why are affordances interesting for robotics ?

Affordance is a useful concept in robotics to formalize the relationship between the robots abilities and its surrounding. To enable a robot to adapt its skills to a current environment, an internal representation of affordances is practical. Moreover, affordances offer a suitable abstraction and definition of an object according to the robot and to the task, it must achieve. For instance, in this dissertation, the concept of object is defined thanks to affordances (see chapter 1).

Moreover, affordances address the *symbol emergence problem*, as affordances are grounded on raw sensorimotors signals and can formalize high-level actions and tasks.

How to formalize affordances ?

To be used in a robotic system, affordances must be concretely and accurately formalized. From the previous section, an affordance is defined as an emergent relational property of the agent-environment system. But a relation between which aspects of the agent and of the environment ? Sahin et al. [2007] proposed to define an affordance as the relation $(effect, (entity, behavior))$, which means a potential $effect$ produced by the application of an agent $behavior$ on an $entity$ in the environment. In this proposition, an affordance is a relationship between a triplet of $effect$, $entity$ and an agent $behavior$. An $effect$ is defined as a change in some states of the environment that are perceivable by the agent. An $entity$ and a $behavior$ are fuzzier concepts: an $entity$ includes any object, part or area of the environment, and a $behavior$ any action or ensemble of actions the agent can do. This is still not accurate enough to build a robotic system.

The main problem is the different scales of abstraction an affordance can represent. For instance, *(roll, (ball, poke))* could correspond to the definition of Sahin et al. [2007]. But so does *(full glass of water, (empty glass of water, filling up a water container)*. When the first triplet is composed of a simple action with an effect directly produced, the second one is more complex as the *filling up of water a container* action could be composed of *grasp*, *bring on a specific place* and *put water in the container* behaviors and the final effect is perceivable only after the last action.

Ellis and Tucker [2000] introduce the notion of *micro-affordances* which allows to disambiguate this issue. They define *micro-affordances* as *potentiated* components of an affordance. To achieve a high-level task associated with one affordance, several micro-affordances would be perceivable. For instance, an agent perceives a graspable object like a mug. The mug is graspable in its whole, but for an agent, there are several ways to actually grasp it. These different grasping possibilities are associated with different micro-affordances.

From this definition, Zech et al. [2017] distinguish micro-affordances and macro-affordances. Micro-affordances are relational properties of low-level sensory features and actions. Macro-affordances are a combination of micro-affordances in relation with a higher-level ability. Equation 7.1 presents the formalization they proposed:

$$\begin{aligned} &Affords\text{–}\phi(feature, ability) \\ &Affords\text{–}\Phi(Affords - \phi_0, .., Affords - \phi_n, ability) \end{aligned} \tag{7.1}$$

These are two relations, for the first line, between a sensory feature and the ability of an agent which corresponds to simple affordances and for the second between other relations and an ability of an agent which corresponds to chained or composite affordances. So, these relations define formally an affordance.

The first line of the formalization corresponds to the formalization of affordances proposed by Chemero. $Feature$ corresponds to preprocessed sensorimotor data perceived by the agent. These $features$ are processed from the raw sensorimotor flow by the agent hardware. An *ability*, or a *skill* is an action primitive or a series of actions primitives that the agent can execute. Finally, a micro-affordance is the relation between a specific feature and a simple ability that would produce a desired effect. A macro-affordance is the relation between a set of micro-affordances and an ability of the agent.

Object action complexes (OACs) are an attempt to formalize sensorimotor experiences of a robotic agent and give them an abstract representation (Krüger et al. [2011a]). OACs can represent high-level actions that are grounded by lower-level action primitives. Thereby, with OACs, Krüger et al. [2011a] address the *symbol emergence problem*.

They define affordances as a relation between a situation and the actions that it allows. Thus, for them, affordances encapsulate, at the same time, one or several objects and the context. This corresponds to macro-affordances.

An OAC is defined as a triplet (E, T, M) :

- E is an identifier for an execution specification, i.e. an action primitive or a sequence of action primitives.
- $T : S \rightarrow S$ is a prediction function defined over an attribute space S. This is a transition function of how the world and the agent will change if the action is successfully executed. The attribute space is a preprocessed representation of the sensorimotor space.
- M a statistical measure of the success of the OAC in a given past time window.

According to this definition, an OAC is a predictive model of the world linked to an action or to what they call a control program. The M measure is either a quality measure of the prediction function T or a probability of success of the action E.

Sensorimotor contingencies (SMC) theory was proposed by O'Regan and Noë [2001] to formalize how animals and humans autonomously develop a perception. They argue that all thinking systems, such as brains or computers, that are equipped with sensors and actuators, build a perception by learning invariants in their sensorimotor space. From this psychology theory, they develop a formalism (Philipona et al. [2003], Laflaquière et al. [2018]) in which SMC are presented as a mapping between sensor and motor spaces built through experiences:

$$s = \phi_e(m) \tag{7.2}$$

Where s is the state of the sensors, m the motors state and ϕ the mapping function parametrized by e which encapsulate the environmental configuration and structure. The robot can infer the mapping function and its parameters only through sensorimotor experiences. Sensorimotor contingencies seem to correspond to micro-affordances.

Finally, an affordance is either a mapping between sensory signals and motor commands, like in SMC theory or in Chemero formalism. Thus, an affordance is a reduction of the complexity in sensorimotor space that relies on invariant identification. Or, an affordance is viewed as a predictive model of the world which, from a sensorimotor input, predicts possible next sensorimotor states, as in OACs.

In this thesis, we consider only low-level affordances which draw a relation between low-level features and simple actions primitives.

How to learn affordances ?

As discussed in section 7.1, interactive perception is linked with learning which features in the environment affords an action. E. Gibson studied how children develop affordances and claimed that learning is "discovering distinctive features and invariant properties of things and events" (Gibson [2000]) and "discovering the information that specifies an affordance" (Gibson [2003]). Learning affordances and building a meaningful segmentation for a robot is about learning "regularities" in its sensorimotor domain. This view is similar to O'Regan and Noë [2001] and their SMCs. Bohg et al. [2017] consider as a necessary condition for interactive perception that interactions reveal regularities in the sensorimotor space.

In robotics, different approaches are proposed to learn a mapping or a predictive model. Three main approaches have actually been proposed to learn affordances (Zech et al. [2017]). In the first kind of approach, a predictive model or a mapping is built from a ground-truth or an annotated dataset (Myers et al. [2015], Achanta et al. [2010], Katz et al. [2014], Varadarajan and Vincze [2012], Kim and Sukhatme [2014]). In this case, a human expert defines the relationship between features and actions and the training is often offline. In the second kind of approach, the relationship between the action and features is learned from demonstrations (Maestre et al. [2017], Kroemer et al. [2012]). A human teacher shows a given action-effect pair to the robot. The third kind of approaches which is the most used one for affordance learning, relies on autonomous exploration based on the interactive perception paradigm (Dang and Allen [2014], Bierbaum et al. [2009], Montesano and Lopes [2009], Popović et al. [2011], Krüger et al. [2011b], Kraft et al. [2010], Kim and Sukhatme [2015], Ŭgur et al. [2007]).

In this chapter, we are interested in learning a perceptual map that represents environment parts affording a specific action (relevance map). This map is learned from local features extracted from supervoxel.

7.2.3 Learning affordances from local features

According to a recent survey (Zech et al. [2017]), among 146 reviewed papers, 104 papers consider learning affordances directly from a meso level, i.e. considering objects as a whole, while only 27 papers consider it from global level, i.e. by considering the whole environment and only 15 papers from a local level. With the global level, considering the whole environment allows the learning system to integrate the context. The context is important to predict or to do recognition of high-level affordances. Most papers on affordance use the meso level because for most actions having a complete model of an object is practical. For instance, for successful grasps, the object states such as orientation and position or shape are important information. Learning affordances at a local level allows the system to perceive them directly, which is in line with Gibson's view. Moreover considering the local level is simpler and is thus suitable to bootstrap the system. From these 15 papers, 11 are interested in linking local descriptors to affordances for quick or direct perception of the possible actions applicable in the present environment. From these papers, 6 are learned from exploration using an interactive perception approach.

In the literature, the problematic of learning affordances from local features using exploration was not studied a lot. This section reviews different groups of works addressing this question. A first group aims at learning several kinds of affordances with supervised learning on annotated datasets, a second one focuses on the object grasping issue and finally, different kinds of works are mentioned.

Some studies use an annotated dataset to train a model of affordance classification and then integrate this model in a robotic framework, as a tool for planning, task solving or manipulation. Myers et al. [2015] study tool use affordances. They train a classifier on superpixels using SLIC. Achanta et al. [2010] have extended it to work on RGB-D images, with features related to shape. Two classifiers are proposed in this work. A first one is called superpixel hierarchical matching which is heavy and slow for prediction. The second one is a structured random forest which is fast for prediction. So, it is adapted for real-time systems. This last classifier is trained offline. AfRob method proposed by Varadarajan and Vincze [2012] is used to classify affordances from 2D images is a deep neural network trained in batch. AfRob is the adaptation of, previously proposed, AfNet, from the same authors to robotics constraints (fast prediction, light computation). Katz et al. [2014] aim at detecting affordances from stacks of objects. With this aim, an SVM linear classifier is used to learn pulling, pushing and grasping affordances. As they use simple shaped objects and only consider their facets as features, i.e. small planar surfaces which compose a 3D shape, they can use a simple linear classifier, especially if trained offline on an annotated dataset. In the same idea, Kim and Sukhatme [2014] proposed a method to detect affordances of surfaces based on a geometrical analysis of the pointcloud, K-means clustering, and logistic regression.

Those methods proposed efficient tools for robotic systems to detect affordances, but they are all based on supervised learning on dataset annotated by a human expert. The annotation is a costly process that naturally limits the learned model to the datasets produced by the expert. Moreover, affordances in ecological psychology depend on the agent body structure and on the actions it is capable of. Another approach is thus to let the robot explores its environment with one or several actions and collects information about the affordances in its surrounding and discovers by itself the affordances.

The works of Dang and Allen [2014], Bierbaum et al. [2009], Montesano and Lopes [2009], Popović et al. [2011], Krüger et al. [2011b], Kraft et al. [2010] are focused on building affordance maps of successful grasp on an object. Bierbaum et al. [2009] let a robotic hand with tactile sensors explores an unknown object in simulation. The robot hand has five fingers with one thumb. The system detects a potential grasp by finding opposite flat surfaces. Then, candidate areas for grasping are determined offline on the basis of geometrical analysis of local shape features. The analysis is a heuristic based on the configuration of the hand used. Montesano and Lopes [2009], from their side, proposed a trial and error process to determine the probability of success of a grasp on parts of an object. Learning is based on local visual features in a Bayesian framework. The robot tries to grasp several times the same object part and, with a Bernoulli-beta distribution based on the successes or failures, the system determines the probability of the graspability of this part. In the same idea Dang and Allen [2014] proposed a system that learns a graspable affordance map on objects but they add what they call semantic constraints. These constraints are designed by a human to force grasping to be compatible with a specific task. In the same way, Popović et al. [2011], Krüger et al. [2011b] use Early Cognitive Vision (ECV, Krüger et al. [2010]) for preliminary image processing to extract features with a stereo camera. The features are edges, contours, textures, and surfaces. The robot tries to grasp different objects and associates ECV's features to successful grasps. But, ECV needs textured or complex objects to work properly.

Those works are conceptually similar to ours: a robotic system explores an environment (here an object) with an action (here grasping) and learns to associate local visual features to successful actions. However, they assume that the system is already able to extract objects from a scene and focus on it to learn grasping. In our work, the robotic system have no notion of objects. The whole environment is considered to learn relevant areas for different affordances. From these areas, object candidates could be extracted as a base for the above-mentioned methods. Thus, these works correspond to a later developmental step with respect to ours.

The works of Ŭgur et al. [2007], Kim and Sukhatme [2015], already described in section 2.4, proposed to learn "traversability" and push affordance with a robot exploring an environment and from local features.

In a more developmental perspective, Paletta et al. [2007] proposed a framework to learn composite affordances by starting from micro-affordances. Their approach is split into 3 steps: first, the robot explores its environment with a reactive behavior, like a grasp reflex, and collects visual data consisting of SIFT. Then, in a second step, basic affordances are learned with simple actions such as pushing or gripping. Finally, in the third step, the robot learns composite affordances based on a combination of the basic action used in the previous step. For instance, this combination of actions allow the robot to achieve stacking. They validate their framework with a mobile robot equipped with a stereo camera and a magnetized end-effector. In a real environment the robot tries to learn to identify objects that are liftable with its magnetized end-effector.

7.3 Method

The method used in this chapter is the same as the one described in chapter 5 apart from some modifications described in the following section.

7.3.1 Affordances Formalisation

In chapter 5, the goal was, for a robot, to learn moveable parts of an environment through an autonomous exploration. The robot was interacting with the environment thanks to a push primitive in order to collect data. In this previous work, the goal was to learn which part affords the pushable ability to objects in an environment. In this chapter, the method is extended to two other affordances : *activable* buttons and *liftable* objects.

The output of the classifier was formalized as a conditional probability of a feature to be part of a class (either moveable or non-moveable). In this study, an action a is associated with an effect e. To integrate different possible actions, the conditional probability of a feature will be associated with an action-effect (a, e). Equation 7.3 described this probability:

$$\begin{split}\phi_{(a,e)}(X) = \\ P(\Delta = (a, e)|W, \Theta, X) = \frac{1 + \Gamma(W_e, \Theta_e, X)}{2 + \Gamma(W_e, \Theta_e, X) + \Gamma(W_{\overline{e}}, \Theta_{\overline{e}}, X)}\end{split} \tag{7.3}$$

Where Γ is a Gaussian mixture function, W_e are the weights associated to the GMM of effect e, Θ_e are the parameters of the MVND of the GMM associated to e, $W = W_e \cup W_{\overline{e}}$, $\Theta = \Theta_e \cup \Theta_{\overline{e}}$ and $\overline{e}$ points out the absence of effect.

Thus, $\phi_{(a,e)}(.)$ represents the probability of an action a to produce an effect e for a certain input feature.

With this formalization, we define *composite affordances* as a composition of one or several affordances thanks to the Bayes' rule.

$$\begin{split}P(\Delta_1|X) = P(\Delta_1|X, \Delta_0)P(\Delta_0|X) \\ \phi_{(a_1,e_1)}(X) = P(\Delta_1 = (a_1, e_1)|W, \Theta, X, \Delta_0 = (a_0, e_0))\phi_{(a_0,e_0)}(X)\end{split} \tag{7.4}$$

Equation 7.4 presents the formal representation of a composite affordance which links an action a_1 and an effect e_1 to an action a_0 and an effect e_0. This proposition means that if the feature X affords the action a_1 by producing the effect e_1 then it affords the action a_0 by producing the effect e_0. In the following text, we say that the probability of X to afford a_1 is *filtered* by the probability of X to afford a_0.

Equation 7.5 presents the general case of a composite affordance as a composition of n other affordances. For this equation to be valid, the component affordances must be independent of each other.

$$\phi_{(a,e)}(X) = P(\Delta = (a,e)|W, \Theta, X) \prod_{i=0}^{n} \phi_{(a_i,e_i)}(X) \tag{7.5}$$

For instance, in this chapter, the probability of something to be liftable is filtered by the probability of something to be pushable. Because we assume that something liftable is also pushable, thus the liftable affordance is composed of the pushable affordance.

7.3.2 Classifier

The classifier used in this chapter is the CMMs introduced in chapter 4, modified after the study described in chapter 6. It was shown in this chapter that the maximization of the loglikelihood to validate splits and merges has a limited impact, we have thus removed it (see section 6.5 and algorithms 3 and 2). Also, the split operation is limited to a maximum number of components per class. The algorithms 4 and 5 are the new split and merge operations used in CMMs for the experiments in this chapter. The rest of the CMMs training algorithm remains unchanged.

Algorithm 4 MERGE algorithm

```
procedure MERGE($C$,l,$M_1$, ..., $M_N$)
    $C' \leftarrow closest\_component(C) \in M_l$        ▷ Search the closest component from $C$ in $M_l$
    if $C \cap C' \neq \emptyset$ then        ▷ If component C intersect with C'
        $\tilde{C} \leftarrow C \cup C'$
        $M_l \leftarrow (M_l \setminus \ C, C') \cup \tilde{C}$
    return $M = \cup_{l=1}^{N} M_l$
```

Algorithm 5 SPLIT algorithm

```
procedure SPLIT($C$,l,$M_1$, ..., $M_N$)
    if $|M_l| < K_{max}$ then        ▷ If the number of components of class $l$ is above $K_{max}$
        return $M = \cup_{l=1}^{N} M_l$        ▷ Then abandon the split
    $C' \leftarrow closest\_component(C) \in M \setminus \{M_l\}$        ▷ Search the closest component from $C$
with a label $\neq l$
    if $C' \cap C \neq \emptyset$ then        ▷ If component C intersect with C'
        $C_1, C_2 = split(C)$
        $M_l \leftarrow (M_l \setminus \{C\}) \cup \{C_1, C_2\}$
    return $M = \cup_{l=1}^{N} M_l$
```

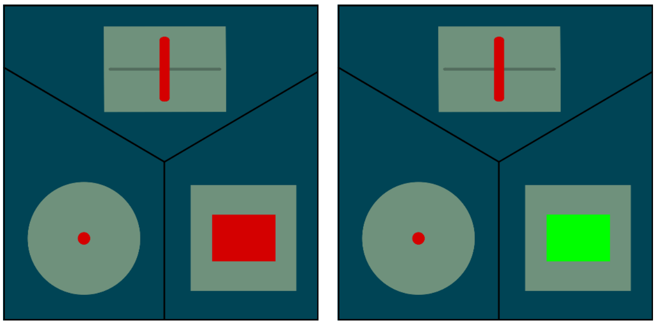
K_{max} is the maximum number of components.

The feature used to train the classifier is the central FPFH and CIELab color histogram described in section 6.4.

7.3.3 Primitives and Effects Detection

Pushable Affordance This affordance corresponds to the one learned in chapter 5. The action primitive and the effect detection remain the same. The action is a push primitive described in section 5.4.5 and the effect detector is a simple detection of change on the chosen supervoxel, as described in the 5.4.6.

Activable Push-Button This affordance is associated with push-buttons which activate a signal displayed on a screen visible to the robot. The action primitive is similar to the push primitive except for the orientation which is only vertical or horizontal in the robot frame. The push primitive used to learn the pushable affordance has a continuous range of orientations. The effect detector is a recognition system which allows the robot to see if a button is pushed. The state of the buttons is displayed on a screen like in the pictures of figure 7.1. The state is perceived by the robot thanks to a visual recognition system implemented with OpenCV. This system is specific to the interface.



(a) No button pushed (b) At least button is pushed

Figure 7.1: Interface which displays on a screen the state of different interactive modules. This interface was developed within the DREAM project. For the present study, only the right bottom part is used. It displays the buttons state. The rectangle is red if no button is pushed and it becomes green if at least one button is pushed.

Liftable Affordance Among the pushable parts in the environment the robot will try to learn liftable parts. It is assumed that liftable parts are first pushable, thus, liftable affordance is a composite affordance, composed by the pushable affordance. The probability to afford the lift primitive is filtered by an already learned probability to afford the push primitive. Therefore, the exploration is biased by a relevance map of pushable affordance.

For this affordance, the robot uses a lift primitive that consists in going above the target, rotating the wrist of its gripper in a certain orientation, then going done and close the gripper before finally going up again and letting the lifted "thing" fall by reopening the gripper. To detect if something is lifted, the opening of the gripper is checked before reopening the gripper. If it is not fully closed, the target will be considered as lifted. In this primitive the gripper is fixed in vertical orientation, thus, only liftable object laying on a horizontal plane is considered here. The approach can be extended to any liftable object with an appropriate lift primitive. In the environment of figure 7.2, the pushable objects are all laying on the horizontal plane. So, the assumption of liftable parts are on a horizontal surface is already contained as a result of the representation of the push relevance map. This is of course, specific to the environment presented here but to generalize to other environments a modification to the lift primitive is enough.

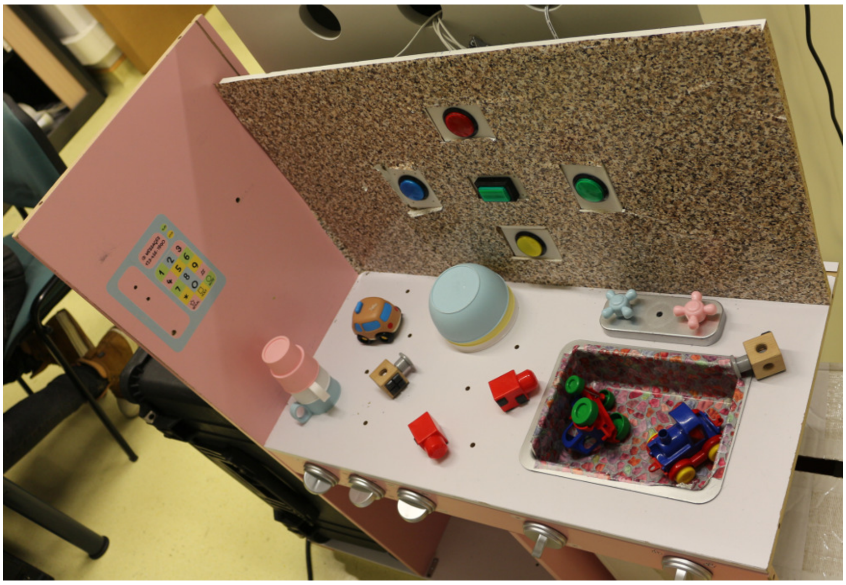
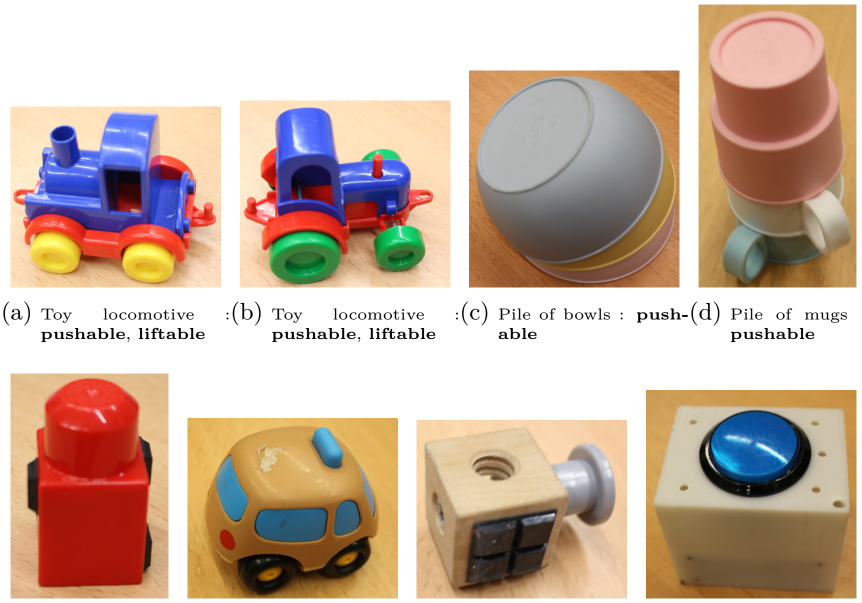


Figure 7.2: The setup used for all the experiments. The setup is a kitchen toy with 5 interactive push buttons integrated into a vertical plane.

7.4 Experiments

For each of the three affordances, 4 experiments have been conducted. The parameter α is fixed to 0.6 and the maximum number of components is fixed to 4 for all experiments. These parameters are fixed according to the experiments done in the previous chapter. The experiments are conducted with the PR2 robot (see figure 5.5) following the protocol in the *real world experiments* described in section 5.5.1.

An initialization step has been added in the experiments of liftable and activable push-button in which the system is forced to gather at least 10 samples of each class. This is achieved by forcing the system to store a new data only if its label corresponds to the class with the less samples. This initial step extend the time of an experiment but allows the system to start from a balanced dataset. With a uniform random sampling, the chance to gather positive samples in these experiments is very low, thus at the beginning of the experiment, the robot collects only negative samples. Adding this step was not useful for the experiments with the push affordance as the probability to gather positive samples is higher.



(a) Toy locomotive : **pushable, liftable**

(b) Toy locomotive : **pushable, liftable**

(c) Pile of bowls : **pushable**

(d) Pile of mugs : **pushable**

(e) Duplo bricks : **pushable, liftable**

(f) Toy car : **pushable**

(g) Wooden cube : **pushable, liftable**

(h) Push-button : **activable**

Figure 7.3: 8 different types of objects used in the experiments. The affordance expected to be linked to these objects is indicated in bold.

Figure 7.3 is a collection of pictures representing the objects used in the experiments: 3 bowls in a pile, 3 mugs in a pile, two different toy locomotives, Duplo bricks, two identical wooden cubes, and 5 push-buttons. Of course, the pile of bowls and mugs (see 7.3c and 7.3d) can be dismantled during an experiment. The Duplo bricks are of different colors (red in the pictures 7.3e): green, red, purple, orange, yellow. There are five push-buttons, all are visible in picture 7.2: circular blue (the one in picture 7.3h), red, yellow, green and squared green. Figure 7.3 indicates in bold for each object its expected affordance.

To assess the performance of the trained classifier, precision, recall, and accuracy are computed by following the methods presented in 5.5.2. These measures are computed according to a ground truth. As explained in chapter 5, the ground truth is obtained from a snapshot of the scene without the objects that afford the studied action which corresponds to the background. For the pushable affordance, the ground truth is exact as it corresponds just to the background and the buttons. For the activable push-buttons, the ground truth is approximative because only a part of the button is activable, the colored central part (see 7.3h) while the ground truth we have set takes into account the whole white box. Thus, the performances should be slightly better than the one presented in the results section. For liftable affordance, the ground truth is even less accurate as it corresponds to our a priori about what the robot may lift or not. An autonomous exploration is interesting and useful precisely when the ground truth is difficult to set. In our case, it is difficult to predict exactly the robustness of the lift according to the robot capacity and the designed lift primitive.

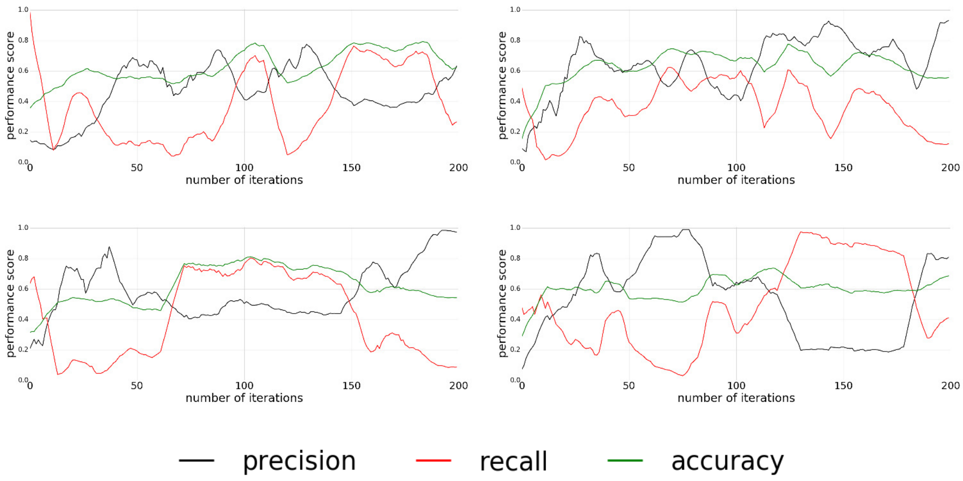


Figure 7.4: Plots of precision, recall, and accuracy for pushable affordance

7.5 Results

For each experiment, the precision, recall, and accuracy scores of each replication are presented separately to avoid losing information.

The precision, recall, and accuracy scores of the experiment for the pushable affordance (presented in figure 7.4) are satisfying considering the complexity of the setup. The classification quality is very different for each replication. In the first experiment (the top left part of the figure 7.4), the classifier converges only around the 150th interaction with an accuracy around 0.8, a recall varying between 0.6 and 0.8, and a low precision around 0.4. Finally, for this replication, the quality drops at the end. For the second and third experiments (the top right and the bottom left parts of the figure 7.4) the classifier converges around the 60th interaction. For the second replication, the accuracy, recall, and precision are not stable and the classifier starts diverging after the 100th interaction. The classifier, of the third replication, converges to an accuracy and a recall around 0.8 and a precision between 0.4 and 0.5 and stays stable. But it diverges after the 150th interactions. For the last replication (the bottom right part of the figure 7.4), it is difficult to isolate a period of convergence of the classifier. The classification quality of this experiment is very unstable.

For all the replications, the quality of classification diverges at the end. The divergence is probably due to mislabeled samples and to splitting or merging components which were not suitable to represent the data. The instability of the classification quality, clearly visible in the second replications, is due to the inconsistency of the supervoxel segmentation when extracted on a video stream as shown in figure 7.5.

The figure 7.5 shows three pictures representing push relevance maps. These relevance maps have been extracted with the same classifier on the same static scene on a video stream. The variability of the relevance map over these three images are due to the extraction of the supervoxel which produces a different segmentation at each frame. The variability of the segmentation is due to the noise of the depth stream. The higher the noise, the higher the variability is. On these pictures, the toy locomotives and the button are the noisiest areas. On these areas, the geometrical features can change a lot, which is due to the variation in the shape of the supervoxels.

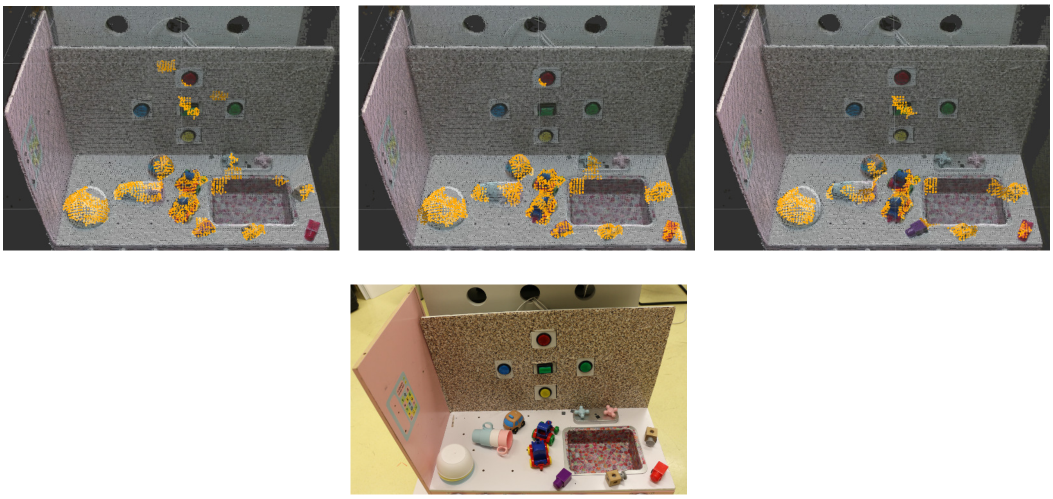


Figure 7.5: Three push relevance maps extracted on the same scene and with the same classifier on a video stream. The differences visible between the three maps is due to the extraction of the supervoxels which produces a different segmentation at each frame. The bottom picture represents the environment on which the relevance maps have been extracted.

The precision, recall, and accuracy scores of the experiment with the push-buttons are shown in the figure 7.6. In this experiment, the replications give also different results. For the first replication (the top left part of the figure 7.6), the classifier converges around the 80th interaction and keep the quality of classification steady around a value of 0.6 for the accuracy and the precision, a value of 0.5 for the recall. For the second replication (top right of the figure 7.6), the classifier converges around the 75th interactions with an accuracy around 0.7, a recall around 0.5 and a precision under 0.4, but this replication starts to diverge around the 160th interaction. For the third replication (the bottom left part of the figure 7.6), the classifier converges quickly to a value between 0.7 and 0.8 for the accuracy, around 0.6 for the recall while the precision increases slowly during all the replication. The accuracy and the recall slowly decrease after the 100th interaction. Finally, the last replication (bottom right of the figure 7.6) presents poor results. The classifier converges first between the 50th and 100th interaction, then diverges, and then converges again to a low classification quality, for finally diverging.

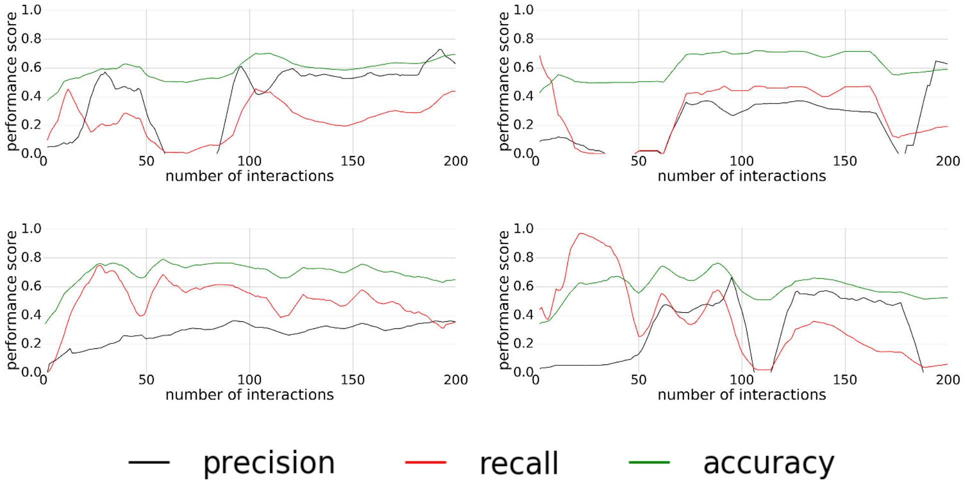


Figure 7.6: Plots of precision, recall, and accuracy for activable push-buttons

Overall, the classification is more stable for this experiment than for the experiments with the pushable affordance. The main difficulty in this experiment is that buttons represent a small area. The size of the actual pushable area is even smaller, about the size of a supervoxel. This introduces noise on the extracted features. A solution may be to reduce the size of the supervoxels, but if a supervoxel contains too few points, the features could be inconsistent. Moreover, this reduced size creates a strong requirement in terms of the accuracy of the action primitive to prevent mislabeling.

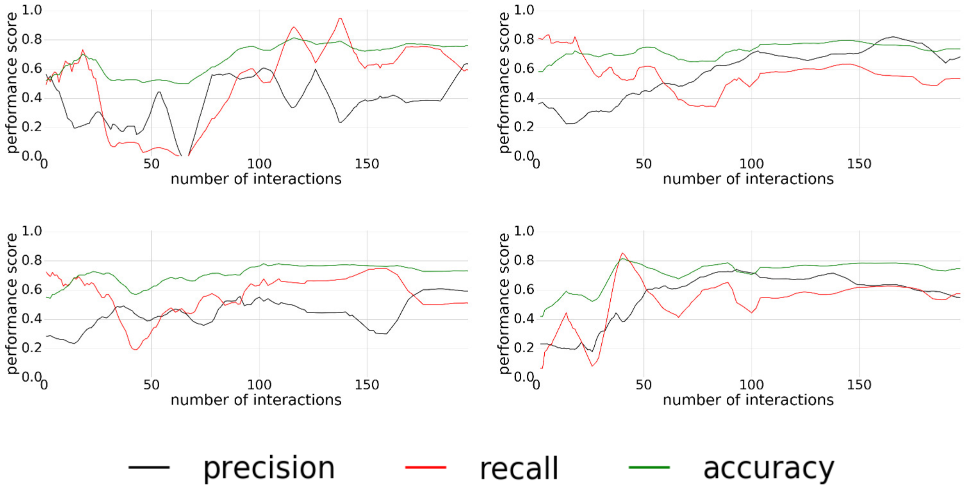


Figure 7.7: Plots of precision, recall, and accuracy for liftable affordance

Figure 7.7 represents the performances monitored during the experiment conducted for the liftable affordances. For the first and the third replications (the left part of figure 7.7), the quality scores have similar shapes, the convergence is reached around the 100th interaction with a low precision and an accuracy, and a recall between 0.7 and 0.8. For the first replication, the recall, and precision are unstable between the 100th and the 150th interactions. In both, the recall and precision cross themselves to have a higher precision than recall which is traduced by a light decreasing of the accuracy. For the second and fourth replications (the right part of figure 7.7), the classifier converges after the 100th interaction, with an accuracy around 0.8, a recall around 0.6, and a higher precision around 0.7. Unlike the two previous experiments (pushable and push-button), the classification quality does not seem to diverge at the end of the experiment, except for the forth replication for which the precision decreases slowly after the 150th interactions.

As in the previous experiment, this experiment gives stable results. The low precision, observed on the first and third replications, is probably due to the inaccuracy of the ground truth. Finally, the stability of the convergence may be due to the use of a push relevance map to filter the classification which does not change during the experiment. However, this does not explain entirely the absence of divergence.

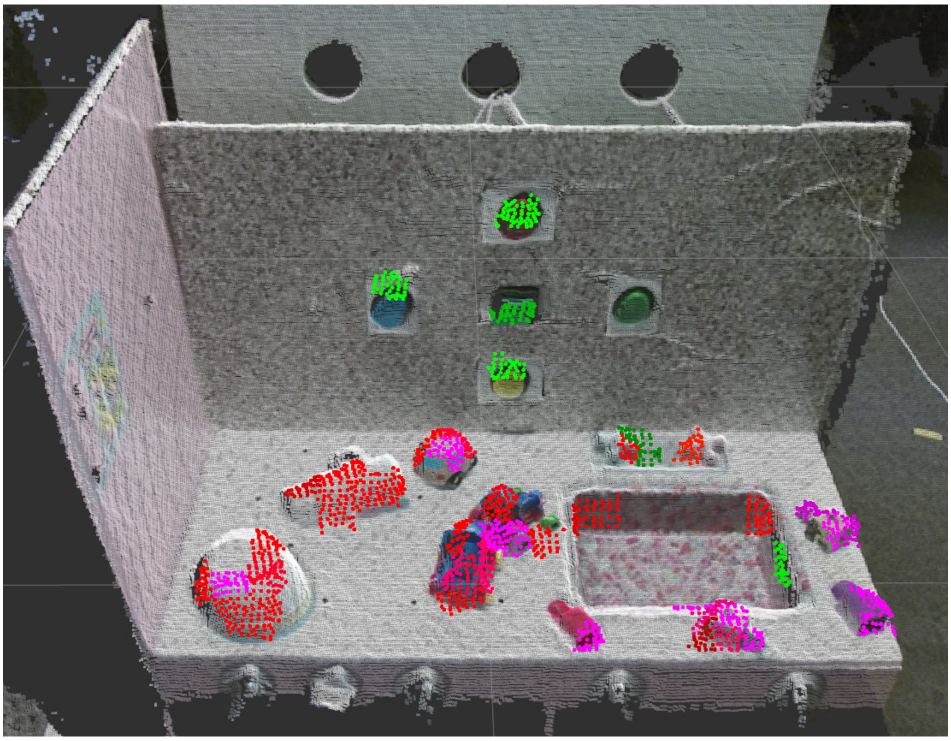
Figure 7.8 represents an affordances map obtained thanks to the experiments described above. This map represents the areas categorized as pushable buttons in green, as liftable objects in yellow, and as pushable objects in orange. It was obtained by selecting the best performing classifier among the experiments and at the best moment inside a replication. Only supervoxels of both relevance maps with a probability equal or higher of 0.5 are displayed.

An interesting property in this affordance map, is the low overlap between the parts predicted to be pushable and to be a push-button. Also, as expected, the pile of bowl is detected as only pushable. The other objects are predicted as pushable and liftable or partially liftable. This affordances map is a proof of concept of what can be obtained with the proposed approach. For each experiment, more replications are required for a better assessment of the method robustness. The instability of the classifier needs to be dealt with for this approach to be more reliable.

7.6 Discussion and Future Works

The experiments described in this chapter are proofs of concept of how the proposed approach could be used to learn affordances map. The results have shown a large variability over the four replications done for each affordance and the classifiers trained for the pushable and activable push-button affordances diverge on the four replications. The stability of the method thus still requires improvement for it to be robust enough in the complex environments it was tested on here.

However, from these experiments, suitable relevance maps have been produced and combined into a meaningful affordances map. The relevance maps of the push-



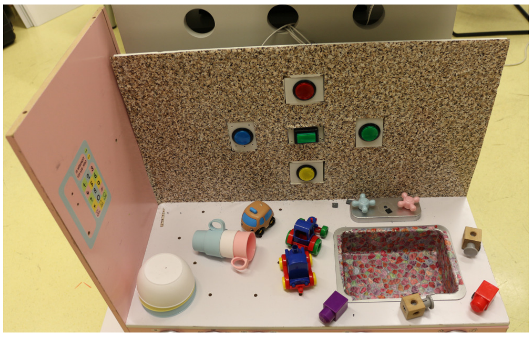


Figure 7.8: Affordances map of liftable activable push-buttons and pushable affordances. Colored areas indicate areas classified with a probability above 0.5, in red to afford the push primitive, in purple to afford the lift primitive and in green to be an activable push-buttons. The bottom picture represents the environment on which the affordances map has been extracted.

button and of pushable affordances do not overlap, which shows the capacity of the classifier to learn different concepts. The classifier is also able to refine a concept as it is shown in the experiment with the liftable affordance.

The generated affordances map represents the ability of action primitives to generate expected effects on each part of the environment. Their precision and success rate is thus critical and the poor accuracy of the button pushing or object lifting primitives probably plays a significant role in the instability of the obtained results.

In a future study, other affordances can be explored like "reachability". For each action primitive a "reachabililty" relevance map could be learned by testing the areas where the robot can apply the action primitive. Then, the classification for the affordance linked to the used primitive will be filtered by the associated "reachability" relevance map. However, to learn "reachability", the proposed framework will have to be extended. Spatial or proprioceptive information should be used as features. But this information is different from the ones used in our method. For instance, if spatial information is combined with FPFH and color histograms, the generalization in term of the spatial position of the objects will be lost. Moreover, the CMMs classifier may not be adapted to learn on spatial or proprioceptive information. Therefore, by keeping the same structure, another classifier may be used with other features.

7.7 Conclusion

In this chapter, the method presented in the previous chapters to produce a relevance map with a push primitive is extended to two other action primitives: a lift primitive and a push button primitive. Also, the formalization is extended to integrate affordances, therefore permitting certain modularity with respect to the action primitive and the effect detector. This formalization includes a representation of simple affordances and composite affordances. The method has successfully built an affordances map, segmenting the visual scene and identifying pushable and liftable objects as well as activable push-buttons on a complex real environment.

Chapter 8

Conclusion and Discussions

Contents

8.1 Summary of the contributions

In this thesis, we have proposed a framework to learn a perceptual map, called *affordances map*, through the autonomous exploration of an environment via interaction. An affordances map is the combination of several *relevance maps* each of which is relative to an affordance. The main goal was to learn such a representation with a minimum of environment specific assumptions. Finally, the approach relies on few hypotheses :

- The action primitive is able to produce effects that can be detected by the effect detector, or, said differently, the effect detector is able to detect the effects produced by the action primitive ;
- The smaller parts of the environment with which the robot can interact are larger than the supervoxels ;
- The background is sufficiently different from the elements that afford the action primitive to be separable by a classifier.

These hypotheses are not specific to a particular environment. However, the method is limited to environments which are adapted to robots with arms and manipulation tasks. For instance, we have not considered mobile robots and navigation tasks.

The framework is composed of 4 different modules: a classifier, a supervoxel features extractor, an action primitive, and an effect detector. These modules are relatively independent and could be changed to learn different representations or increase the performance. The classifier, called *collaborative mixture models (CMMs)*, is trained online, based on Gaussian mixture models with an unknown number of components and uses a query strategy based on uncertainty reduction to be sample efficient. The affordances map is based on supervoxel segmentation. So, the smallest 3D visual components considered are supervoxels. All the visual features computed for the training are characterizing supervoxels. Different visual features have been tested and finally, the feature with the best performance is based on FPFH descriptor and CIELab color histograms. The chosen action primitive and effect detector determine which affordance will be learned. In this thesis, pushable, liftable and activable push-button affordances have been tested. Pushable affordance is associated to a push primitive and a detection of a change in the image; liftable affordance to a lift primitive and a gripper position monitoring; and activable push-button to a push button primitive and a detection of a state displayed on a screen. The method has been tested on the PR2 robot using its two arms.

The conducted experiments demonstrate the capacity of the method to produce usable affordances map. Usable because these maps segment precisely enough the objects or relevant elements of the environment to bootstrap other learning steps. Moreover the segmented elements could be placed where ever in the environment and be still segmented. However, the approach presents some limitations discussed in the next section.

8.2 Discussion and Limitations

8.2.1 CMMs Limitations

CMMs being based on GMM its computational time for prediction increases as the dimension of the feature space grows. Computing an MVND alone is heavy and a GMM is a sum of this distribution. Moreover, the intersection condition defined in section 3.2.2 introduces the constraint of a minimum number of samples in a component to be split or merged. The component must contain at least as many samples as the dimension of the feature space. Therefore, the feature space cannot be too large otherwise too many samples would need to be collected and thus, CMMs would not be sample efficient. This drawback prevents from using features built by deep neural networks because the dimensionality of convolutional layers are too large. Convolutional Neural Network is the most used for image processing Krizhevsky et al. [2012].

The main limitation of CMMs is the need to determine a budget for its training. This parameter is naturally dependent on the environment complexity because the more complex the environment, the more samples need to be gathered to have a representative dataset. Also, a maximum number of components was introduced in chapter 6 which is linked to the budget. This parameter is also dependent on the complexity of the environment. Moreover, during an exploration, the classification quality of CMMs is not guaranteed to be maintained as samples are added. An ending criterion is needed to finish the training at a maximum quality of classification. But online learning makes the definition of an intrinsic quality measurement difficult.

8.2.2 Supervoxels

Supervoxels bring a lot of advantages. The segmentation into supervoxels using the VCCS methods respects the boundaries of objects by taking into account the depth information. Thus, a supervoxel is not crossing two objects or the object and the background which is very practical for classification. However, this method brings some limitations. First, supervoxels are used in this thesis as a downsampling method, so, the relevant components of the environment should not be smaller than a supervoxel. This introduces a limitation on the scale, if an object or component is too small, or too far from the camera, a supervoxel can overlap it or it can be segmented into only one supervoxel and our method relies on the assumption that an object is segmented into several supervoxels.

The second drawback is the inconsistency of the segmentation over frames. When supervoxels are extracted over a video stream, the segmentation may be different at each frame. The more complex and the noisier the depth information, the more inconsistent the segmentation over the frames is. This issue prevents supervoxels to be tracked. This issue could be solved by using the enhanced VCCS method proposed by Papon et al. [2013b] which features permanent supervoxel segmentation.

8.2.3 Learning from local features

In the proposed approach, the system learns only from local features. Considering only the local level has allowed our system to learn simple representations that require little prior knowledge. However, this choice creates some limitations on the level of abstraction of the affordances which the system can learn. For instance, learning *stackable* objects with the proposed method would be difficult because knowing if an object is stackable over another requires information such as their sizes or shapes. Moreover, it is not straightforward to detect if an object is successfully stacked without the model of the objects. To learn high-level affordances, a system needs information about the context, the environment structure or the objects. Therefore, other kinds of features are needed such as relational features, which give information about the relation between the objects in a scene, or global geometric descriptors, which characterize the global shape of an object.

The liftable affordance presented in chapter 7 is on the line between low-level and high-level affordances because of the lift primitive implemented for the experiments. This primitive has a high probability of success with small objects for which it will be successful for any part of it. But with bigger objects, the failure probability is higher. The shape of objects would be a piece of information of great help to increase its chance of success.

Moreover, action primitives and effect detectors greatly condition what will be learned by the system. Of course, they define which affordance is learned, but they can also introduce limitations. The classifier needs the primitive to have a sufficiently low failure rate to separate the elements which afford the action and those do not afford. Thus, mistakes into the designed action primitive could lead to failed experiments. Of course, more complex the action primitive, more failures can occur. Thus, the proposed method should be more robust with simple action primitive.

Thus, our method is limited to learn low-level affordances which link a simple action to a direct effect, in other words, it is limited to micro-affordances. And the effect must be easily detectable with little knowledge about the environment. This limitation mainly stems from the context we have chosen in which objects are not known yet. The approach could be extended to a later stage in development in which primitives and effect rely on objects that the classifier knows nothing about. It could be useful to affordances associated with parts of objects, like the handle of a cup for instance.

8.3 Future Works

8.3.1 Possible Improvements

As already mentioned, the supervoxel segmentation is inconsistent over the frames. Using a method of oversegmentation that produces consistent segments would significantly enhance our method. This would permit to implement real-time effect detectors by tracking the segments like in the work of van Hoof et al. [2014]. Moreover, collecting data would be more efficient as the segments could be memorized and a world-model could be built as in the work of Papon et al. [2013b].

Another important area for improvement is the classifier. As already discussed, there is room for improvement on CMMs. There are two priorities: first, implementing an "unlearning" mechanism and then implementing an unsupervised quality measure as an objective function for the classifier.

Unlearning in machine learning consists in erasing parts of the model considered as outdated. Thus, an unlearning mechanism would allow the classifier to be less dependent on the samples gathered at the beginning and to be more adaptive to changes in the environment. Unlearning in GMM has not been studied a lot. Kristan et al. [2008] proposed a process of unlearning based on *attenuation* of a distribution to be combined with a new distribution.

To implement an objective function for the classifier, here are the most promising quality measures :

- The *loglikelihood* is the objective function used in expectation-maximization algorithm. Even if, in this thesis, the loglikelihood has been removed from the algorithm (see section 6.5), it can be used in another way.

- The *model variance* is already described and discussed in section 6.5. Thanks to the property written in equation 6.2, the variance of the model provides interesting information about the noise due to the data acquisition and the error of the model.

- The *Kullback-Leibler divergence* is defined in equation 6.3 and described in section 6.5. This measure is interesting as it gives a quantification of the difference between two distributions that is linked to entropy and loglikelihood.
- A measure of the uncertainty of the model. The query strategy of CMMs is already based on uncertainty measurement, but this information could be more exploited.

Finally, it would be interesting to test the framework with other classifiers. The most promising ones proposed in the literature are the kernel density estimation with a Gaussian kernel proposed by Kristan et al. [2011] and the random forest proposed by Saffari et al. [2009]. Also, a simple linear SVM could be used as a baseline for a comparison of the different mentioned classifier. To this end, LASVM proposed by Bordes et al. [2005] is a promising candidate. But all of these machine learning algorithms are missing an important feature which is a query strategy. To be used in our framework, a query strategy should be added to each of them.

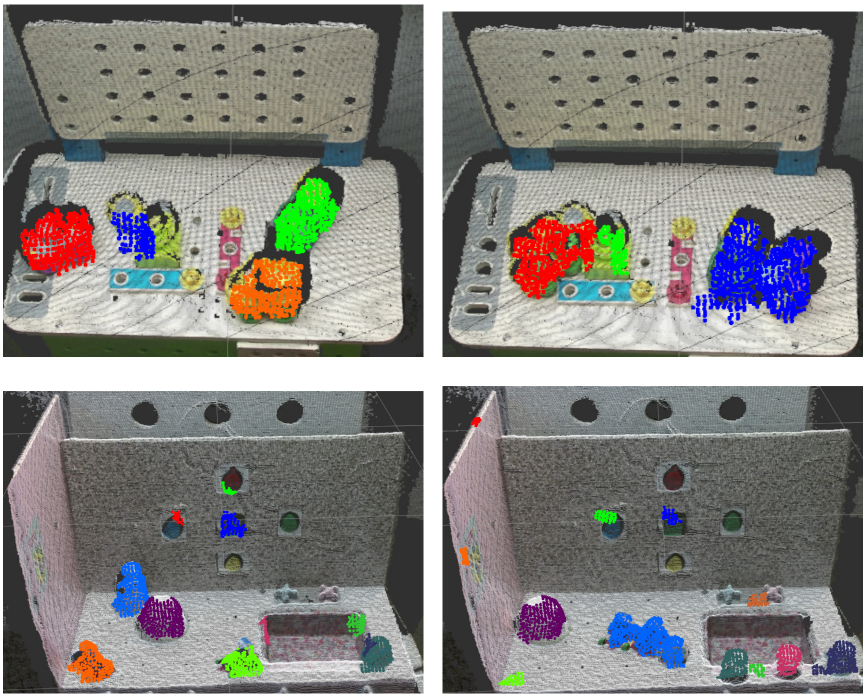


Figure 8.1: Four pictures of pointclouds with object candidates colored. The pictures show different positions of the objects on the workbench setup, in top pictures, and on the kitchen setup, in the bottom pictures. These object candidates have been extracted from a different push relevance map for each setup. The colors are randomly picked.

8.3.2 Next Developmental Steps

The approach proposed here aims at being a bootstrap step for other developmental steps. As shown in figure 5.2, interactive perception methods rely on a bootstrap phase in which passive image processing techniques are applied to produce a segmentation representing either object candidate or parts of objects. This phase introduces assumptions about the environment or objects structure and can be replaced by our approach. It removes a lot of assumptions and thus makes these approaches relevant in a larger range of environments.

Indeed, relevance maps can be used to produce objects candidates. For instance, figure 8.1 shows four extractions of object candidates based on a push relevance map and applied on two different setups: the workbench on the top pictures, and the kitchen on the bottom pictures. For the workbench, on both pictures, the same classifier is used, and moveable objects are in different places. It is the same for the pictures of the kitchen. The left pictures show the ideal case when all objects are well separated. In this case, the objects are already segmented. In both right pictures, the blue object candidates include for several objects because their are adjacent. In this case, another interactive perception process is needed to separate the objects as the approaches proposed by van Hoof et al. [2014], Gupta and Sukhatme [2012], Chang et al. [2012], Eitel et al. [2017].

In the DREAM project workflow, the next developmental step will be a babbling focused on objects with the aim to collect visual data such as RGB, depth images and proprioceptive information from the robot. From these information, models of the objects could be learned offline using deep learning for instance (Wu et al. [2015]). With 3D models of objects, the robot can learn to manipulate them. Within the DREAM project, an evolutionary algorithm method is proposed called *quality diversity search*. In simulation, the system tries a great number of behaviors to achieve a specific goal, for instance, throwing a ball (Kim et al. [2019]).

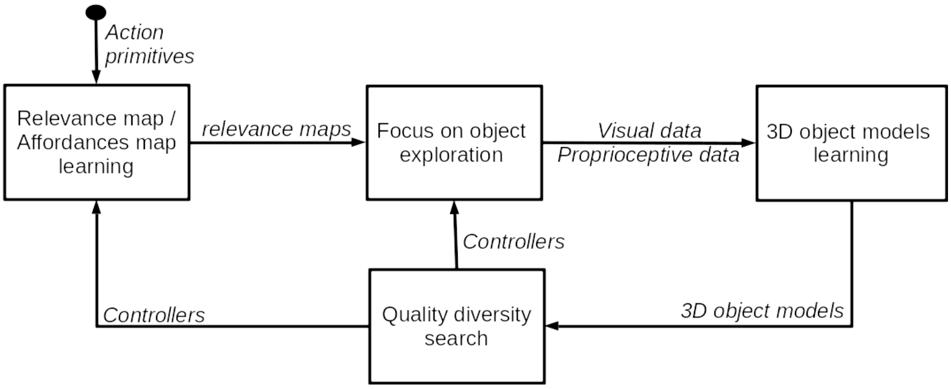


Figure 8.2: Example of developmental steps proposed within the DREAM Project. Each box represents a developmental process, the arrows represent data transfers. The black dot represents the starting point.

Figure 8.2 shows an example of the developmental steps proposed within the

DREAM project. First, the system learns a relevance map or an affordances map with basic action primitives and effect detectors. Then, the system learns 3D models of objects by alternatively gathering data by interacting with the objects thanks to the relevance map and processing these data and building object models. Finally, a quality diversity search is applied to the object models. The new controllers learned thanks to quality diversity search could be reused by our method to build new relevance maps.

Bibliography

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, P Fua, and S Susstrunk. SLIC Superpixels. *EPFL Technical Report 149300*, (June):15, 2010. ISSN 149300. doi: 10.1109/TPAMI.2012.120. (Cité en pages 19, 101 et 102.)

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. (Cité en page 21.)

Niklas Bergström, Carl Henrik Ek, Mårten Björkman, and Danica Kragic. Scene understanding through autonomous interactive perception. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6962 LNCS:153–162, 2011. ISSN 03029743. doi: 10.1007/978-3-642-23968-7{_}16. (Cité en pages 49, 51 et 52.)

Christian Bersch, Dejan Pangercic, Sarah Osentoski, Karol Hausman, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Segmentation of Textured and Textureless Objects through Interactive Perception. *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, (July), 2012. (Cité en pages 51 et 52.)

Alexander Bierbaum, Matthias Rambow, Tamim Asfour, and Rüdiger Dillmann. Grasp affordances from multi-fingered tactile exploration using dynamic potential fields. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 168–174. IEEE, 2009. (Cité en pages 101 et 103.)

Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6): 1273–1291, 2017. (Cité en pages ix, xiii, 1, 7, 10, 49 et 101.)

Antoine Bordes and Léon Bottou. The huller: a simple and efficient online svm. In *ECML*, pages 505–512. Springer, 2005. (Cité en pages 27 et 29.)

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6 (Sep):1579–1619, 2005. (Cité en pages 12, 27, 29, 30 et 119.)

Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Salient Object Detection: A Survey. pages 1–26, 2014. ISSN 1941-0042. doi: 10.1109/TIP.2015.2487833. URL http://arxiv.org/abs/1411.5878. (Cité en pages x, xiv, 2 et 54.)

Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Salient Object Detection: A Benchmark. pages 1–15, 2015. ISSN 16113349. doi: 10.1109/TIP.2015.2487833. URL http://arxiv.org/abs/1501.02741. (Cité en pages 56, 57 et 63.)

Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998. (Cité en page 30.)

Wenbin Cai, Ya Zhang, and Jun Zhou. Maximizing expected model change for active learning in regression. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 51–60. IEEE, 2013. (Cité en page 38.)

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010. (Cité en page 21.)

Angelo Cangelosi, Matthew Schlesinger, and Linda B Smith. *Developmental robotics: From babies to robots.* MIT Press, 2015. (Cité en pages xii, 5 et 6.)

Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009. (Cité en pages 27 et 31.)

Marisa Carrasco. Visual attention: The past 25 years. *Vision research*, 51(13): 1484–1525, 2011. (Cité en pages ix et 2.)

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in neural information processing systems*, pages 409–415, 2001. (Cité en pages 27 et 29.)

David J Chalmers, Robert M French, and Douglas R Hofstadter. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental & Theoretical Artificial Intelligence*, 4(3):185–211, 1992. (Cité en pages xii et 4.)

Lillian Chang, Joshua R Smith, and Dieter Fox. Interactive singulation of objects from a pile. *Proceedings - IEEE International Conference on Robotics and Automation*, (May):3875–3882, 2012. ISSN 10504729. doi: 10.1109/ICRA.2012.6224575. (Cité en pages 49, 51 et 120.)

Krishneel Chaudhary, Chi-Wun Au, Wesley P Chan, Kotaro Nagahama, Hiroaki Yaguchi, Kei Okada, and Masayuki Inaba. Retrieving unknown objects using robot in-the-loop based interactive segmentation. In *System Integration (SII), 2016 IEEE/SICE International Symposium on*, pages 75–80. IEEE, 2016. (Cité en page 51.)

Anthony Chemero. An outline of a theory of affordances. *Ecological psychology*, 15 (2):181–195, 2003. (Cité en pages xv et 97.)

C. Craye, David Filliat, and J.-F. Goudou. Exploration Strategies for Incremental Learning of Object-Based Visual Saliency. *Proc. of the 5th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*, pages 13–18, 2015. doi: 10.1109/DEVLRN.2015.7346099. (Cité en pages 30 et 31.)

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503, 2015. (Cité en page 7.)

Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005. (Cité en page 35.)

Hao Dang and Peter K Allen. Semantic grasping: planning task-specific stable robotic grasps. *Autonomous Robots*, 37(3):301–316, 2014. (Cité en pages 101 et 103.)

Arnaud Declercq and Justus H Piater. Online learning of gaussian mixture models-a two-level approach. In *VISAPP (1)*, pages 605–611, 2008. (Cité en pages 31, 32, 34 et 35.)

S. Doncieux. Creativity: A driver for research on robotics in open environments. *Intellectica*, 2016/1(65):205–219, 2016. (Cité en page 53.)

Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E Gusz Eiben. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2:4, 2015. (Cité en page 7.)

Stephane Doncieux, David Filliat, Natalia Díaz-Rodríguez, Timothy Hospedales, Richard Duro, Alexandre Coninx, Diederik M Roijers, Benoît Girard, Nicolas Perrin, and Olivier Sigaud. Open-ended learning: A conceptual framework based on representational redescription. *Frontiers in neurorobotics*, 12, 2018. (Cité en page 8.)

Andreas Eitel, Nico Hauff, and Wolfram Burgard. Learning to singulate objects using a push proposal network. *arXiv preprint arXiv:1707.08101*, 2017. (Cité en pages 51, 52 et 120.)

Rob Ellis and Mike Tucker. Micro-affordance: The potentiation of components of action by seen objects. *British journal of psychology*, 91(4):451–471, 2000. (Cité en page 99.)

Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. (Cité en page 51.)

Paul Fitzpatrick and Giorgio Metta. Grounding vision through experimental manipulation. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 361(1811):2165–2185, 2003. ISSN 1364-503X. doi: 10.1098/rsta.2003.1251. (Cité en pages 49 et 51.)

Paul M Fitzpatrick and Giorgio Metta. Towards manipulation-driven vision. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 43–48. IEEE, 2002. (Cité en page 49.)

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992. (Cité en pages 38 et 92.)

Eleanor J Gibson. Perceptual learning in development: Some basic concepts. *Ecological Psychology*, 12(4):295–302, 2000. (Cité en pages ix, 1 et 101.)

Eleanor J Gibson. The world is so full of a number of things: On specification and perceptual learning. *Ecological psychology*, 15(4):283–287, 2003. (Cité en page 101.)

James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 1979. (Cité en pages xv, 11, 95 et 96.)

James Jerome Gibson. The senses considered as perceptual systems. 1966. (Cité en pages ix, xv, 2, 11, 95 et 96.)

Megha Gupta and Gaurav S Sukhatme. Using manipulation primitives for brick sorting in clutter. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3883–3889. IEEE, 2012. (Cité en pages 49, 51, 52 et 120.)

Mohamed Farouk Abdel Hady and Friedhelm Schwenker. Semi-supervised learning. In *Handbook on Neural Information Processing*, pages 215–239. Springer, 2013. (Cité en page 9.)

Karol Hausman, Ferenc Balint-benczedi, Dejan Pangercic, Zoltan-csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Tracking-based Interactive Segmentation of Textureless Objects. In *Robotics and Automation (ICRA)*, 2013. (Cité en pages 51 et 52.)

Tucker Hermans, James M Rehg, and Aaron Bobick. Guided pushing for object singulation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4783–4790. IEEE, 2012. (Cité en pages 49 et 51.)

Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008. (Cité en page 36.)

Thomas E Horton, Arpan Chakraborty, and Robert St Amant. Affordances for robots: a brief survey. *AVANT. Pismo Awangardy Filozoficzno-Naukowej*, 2: 70–84, 2012. (Cité en page 98.)

Sheng-Jun Huang and Zhi-Hua Zhou. Active query driven by uncertainty and diversity for incremental multi-label learning. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1079–1084. IEEE, 2013. (Cité en pages 36 et 44.)

Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews. Neuroscience*, 2(3):194–203, 2001. ISSN 1471-003X. doi: 10.1038/35058500. (Cité en pages x, 2 et 54.)

W James. The principles of psychology, vol. 2. ny, us: Henry holt and company, 1890. (Cité en pages xii et 4.)

Lorenzo Jamone, Emre Ugur, Angelo Cangelosi, Luciano Fadiga, Alexandre Bernardino, Justus Piater, and José Santos-Victor. Affordances in psychology, neuroscience, and robotics: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):4–25, 2016. (Cité en page 98.)

Marija Jegorova, Stéphane Doncieux, and Timothy Hospedales. Generative adversarial policy networks for behavioural repertoire. *arXiv preprint arXiv:1811.02945*, 2018. (Cité en page 8.)

Yangqing Jia and Mei Han. Category-independent object-level saliency detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1761–1768, 2013. (Cité en page 56.)

Bowen Jiang, Lihe Zhang, Huchuan Lu, Chuan Yang, and Ming-Hsuan Yang. Saliency detection via absorbing markov chain. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1665–1672, 2013a. (Cité en pages 54 et 55.)

Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2083–2090, 2013b. (Cité en pages 54, 57 et 75.)

Peng Jiang, Haibin Ling, Jingyi Yu, and Jingliang Peng. Salient region detection by ufo: Uniqueness, focusness and objectness. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1976–1983, 2013c. (Cité en pages 55 et 56.)

Zhuolin Jiang and Larry S Davis. Submodular salient region detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2043–2050, 2013. (Cité en pages 55 et 56.)

Dov Katz, Arun Venkatraman, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots*, 37(4):369–382, 2014. (Cité en pages 101 et 102.)

Jacqueline Kenney, Thomas Buckley, and Oliver Brock. Interactive segmentation for manipulation in unstructured environments. *2009 IEEE International Conference on Robotics and Automation*, 2009. ISSN 1050-4729. doi: 10.1109/ROBOT.2009.5152393. (Cité en page 51.)

David Inkyu Kim and Gaurav S Sukhatme. Semantic labeling of 3d point clouds with object affordance for robot manipulation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5578–5584. Citeseer, 2014. (Cité en pages 101 et 102.)

David Inkyu Kim and Gaurav S Sukhatme. Interactive affordance map building for a robotic task. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4581–4586. IEEE, 2015. (Cité en pages 12, 101 et 103.)

Jiwhan Kim, Dongyoon Han, Yu-Wing Tai, and Junmo Kim. Salient region detection via high-dimensional color transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 883–890, 2014. (Cité en page 54.)

Seungsu Kim, Alexandre Coninx, and Stephane Doncieux. From exploration to control: learning object manipulation skills through novelty search and local adaptation. *arXiv preprint arXiv:1901.00811*, 2019. (Cité en pages 8 et 120.)

Dirk Kraft, Renaud Detry, Nicolas Pugeault, Emre Baseski, Frank Guerin, Justus H Piater, and Norbert Kruger. Development of object and grasping knowledge by robot exploration. *IEEE Transactions on Autonomous Mental Development*, 2 (4):368–383, 2010. (Cité en pages 101 et 103.)

Matej Kristan, Danijel Skocaj, and Aleš Leonardis. Incremental learning with gaussian mixture models. In *Computer Vision Winter Workshop*, pages 25–32, 2008. (Cité en pages 27, 31, 32, 34, 35, 93 et 118.)

Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, 44(10-11):2630–2642, 2011. (Cité en pages 27, 31, 32, 34, 35 et 119.)

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. (Cité en page 116.)

Oliver Kroemer, Emre Ugur, Erhan Oztop, and Jan Peters. A kernel-based approach to direct action perception. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2605–2610. IEEE, 2012. (Cité en page 101.)

Norbert Krüger, Nicolas Pugeault, Emre Baseski, LBW Jensen, S Kalkan, D Kraft, JB Jessen, F Pilz, A Kjaer-Nielsen, M Popovic, et al. Early cognitive vision as a front-end for cognitive systems. In *ECCV 2010 Workshop on "Vision for Cognitive Tasks*, 2010. (Cité en page 103.)

Norbert Krüger, Christopher Geib, Justus Piater, Ronald Petrick, Mark Steedman, Florentin Wörgötter, Aleš Ude, Tamim Asfour, Dirk Kraft, Damir Omrčen, et al. Object–action complexes: Grounded abstractions of sensory–motor processes.

Robotics and Autonomous Systems, 59(10):740–757, 2011a. (Cité en pages xv et 100.)

Norbert Krüger, Mila Popovic, Leon Bodenhagen, Dirk Kraft, and Frank Guerin. Grasp learning by means of developing sensorimotor schemas and generic world knowledge. In *AISB Convention*, pages 23–31. Citeseer, 2011b. (Cité en pages 101 et 103.)

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. (Cité en page 93.)

Eva Stergaršek Kuzmič and Aleš Ude. Object segmentation and learning through feature grouping and manipulation. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 371–378. IEEE, 2010. (Cité en pages 51 et 53.)

E Lachat, H Macher, MA Mittet, T Landes, and P Grussenmeyer. First experiences with kinect v2 sensor for close range 3d modelling. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):93, 2015. (Cité en page 57.)

Alban Laflaquière, J Kevin O'Regan, Bruno Gas, and Alexander Terekhov. Discovering space-grounding spatial topology and metric regularity in a naive agent's sensorimotor experience. *Neural Networks*, 2018. (Cité en page 100.)

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010. (Cité en pages xviii et 78.)

Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011. (Cité en page 7.)

Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Belson, David M Bryson, Nick Cheney, et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453*, 2018. (Cité en page 7.)

Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2290–2297, 2009. (Cité en page 19.)

David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning Proceedings 1994*, pages 148–156. Elsevier, 1994. (Cité en page 35.)

Xiaohui Li, Huchuan Lu, Lihe Zhang, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via dense and sparse reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2976–2983, 2013. (Cité en pages 54 et 55.)

Xin Li and Yuhong Guo. Active learning with multi-label svm classification. In *IJCAI*, pages 1479–1485, 2013. (Cité en page 36.)

Risheng Liu, Junjie Cao, Zhouchen Lin, and Shiguang Shan. Adaptive partial differential equation learning for visual saliency detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3866–3873, 2014. (Cité en page 55.)

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. (Cité en page 21.)

Carlos Maestre, Ghanim Mukhtar, Christophe Gonzales, and Stephane Doncieux. Iterative affordance learning with adaptive action generation. In *International Conference on Development and Learning (ICDL) and the International Conference on Epigenetic Robotics (EpiRob)*, 2017. (Cité en page 101.)

A Maleki. Graph-based visual saliency model using background color. (2012), 2017. (Cité en page 54.)

Naoki Abe Hiroshi Mamitsuka et al. Query learning strategies using boosting and bagging. In *Machine learning: proceedings of the fifteenth international conference (ICML'98)*, volume 1. Morgan Kaufmann Pub, 1998. (Cité en page 37.)

Leonard S Mark. Eyeheight-scaled information about affordances: A study of sitting and stair climbing. *Journal of experimental psychology: human perception and performance*, 13(3):361, 1987. (Cité en page 97.)

Stephen Marsland. Machine learning: an algorithmic perspective. chapter 1. Chapman and Hall/CRC, 2011. (Cité en page 8.)

Giorgio Metta and Paul Fitzpatrick. Early integration of vision and manipulation. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 2703–vol. IEEE, 2003. (Cité en page 49.)

Huaqing Min, Chang'an Yi, Ronghua Luo, Jinhui Zhu, and Sheng Bi. Affordance research in developmental robotics: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 8(4):237–255, 2016. (Cité en page 98.)

Luis Montesano and Manuel Lopes. Learning grasping affordances from local visual descriptors. In *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, pages 1–6. IEEE, 2009. (Cité en pages 101 et 103.)

Austin Myers, Ching Lik Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance detection of tool parts from geometric features. In *ICRA*, pages 1374–1381, 2015. (Cité en pages 101 et 102.)

Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998. (Cité en page 93.)

Don Norman. *The design of everyday things: Revised and expanded edition*. Constellation, 2013. (Cité en pages xv et 97.)

J Kevin O'Regan and Alva Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5):939–973, 2001. (Cité en pages ix, xv, 1, 100 et 101.)

Stefan Otte, Johannes Kulick, Marc Toussaint, and Oliver Brock. Entropy-based strategies for physical exploration of the environment's degrees of freedom. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 615–622. IEEE, 2014. (Cité en page 36.)

Pierre-Yves Oudeyer. Intelligent adaptive curiosity: a source of self-development. 2004. (Cité en pages x, 2, 7, 30 et 37.)

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009. (Cité en pages x, 2, 7 et 37.)

Nikunji C Oza. Online bagging and boosting. 2005. (Cité en pages 27 et 30.)

Lucas Paletta, Gerald Fritz, Florian Kintzler, Jörg Irran, and Georg Dorffner. Perception and developmental learning of affordances in autonomous robots. In *Annual Conference on Artificial Intelligence*, pages 235–250. Springer, 2007. (Cité en page 103.)

Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation - Supervoxels for point clouds. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013a. ISSN 10636919. doi: 10.1109/CVPR.2013.264. (Cité en pages xvi, 19 et 20.)

Jeremie Papon, Tomas Kulvicius, Eren Erdal Aksoy, and Florentin Wörgötter. Point cloud video object segmentation using a persistent supervoxel world-model. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3712–3718. IEEE, 2013b. (Cité en pages 117 et 118.)

Timothy Pattent, Micheal Zillich, and Markus Vincze. Action selection for interactive object segmentation in clutter. In *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*. IEEE, 2018. (Cité en page 51.)

Houwen Peng, Bing Li, Rongrong Ji, Weiming Hu, Weihua Xiong, Congyan Lang, et al. Salient object detection via low-rank and structured sparse matrix decomposition. In *AAAI*, pages 796–802, 2013. (Cité en page 55.)

David Philipona, J Kevin O'Regan, and J-P Nadal. Is there something out there? inferring space from sensorimotor dependencies. *Neural computation*, 15(9):2029–2049, 2003. (Cité en page 100.)

Jean Piaget and Margaret Cook. *The origins of intelligence in children*, volume 8. International Universities Press New York, 1952. (Cité en page 6.)

Mila Popović, Gert Kootstra, Jimmy Alison Jørgensen, Danica Kragic, and Norbert Krüger. Grasping unknown objects using an early cognitive vision system for general scene understanding. *IEEE International Conference on Intelligent Robots and Systems*, pages 987–994, 2011. ISSN 2153-0858. doi: 10.1109/IROS.2011.6048619. (Cité en pages 101 et 103.)

Antonin Raffin, Ashley Hill, René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. S-rl toolbox: Environments, datasets and evaluation metrics for state representation learning. *arXiv preprint arXiv:1809.09369*, 2018. (Cité en page 8.)

Sylvia Richardson and Peter J Green. On bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society: series B (statistical methodology)*, 59(4):731–792, 1997. (Cité en pages 33 et 34.)

Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448, 2001. (Cité en page 38.)

Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. `http://pointclouds.org/`. (Cité en pages 20 et 60.)

Radu Bogdan Rusu, Nico Blodow, and Michael 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. (Cité en pages xvi, 20, 21, 22, 23, 24 et 59.)

Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400. IEEE, 2009. (Cité en pages 27, 30, 92 et 119.)

Amir Saffari, Martin Godec, Thomas Pock, Christian Leistner, and Horst Bischof. Online multi-class lpboost. In *Computer Vision and Pattern Recognition*

(CVPR), 2010 IEEE Conference on, pages 3570–3577. IEEE, 2010. (Cité en pages 27 et 30.)

E. Sahin, M. Cakmak, M. R. Dogar, E. Ugur, and G. Ucoluk. To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control. *Adaptive Behavior*, 15(4):447–472, 2007. ISSN 1059-7123. doi: 10.1177/1059712307084689. (Cité en pages 12, 98 et 99.)

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001. (Cité en page 36.)

David Schiebener, Aleš Ude, Jun Morimoto, Tamim Asfour, and Rüdiger Dillmann. Segmentation and learning of unknown objects through physical interaction. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 500–506. IEEE, 2011. (Cité en pages 49, 51 et 53.)

David Schiebener, Aleš Ude, and Tamim Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4959–4966, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6907586. (Cité en pages 51 et 53.)

Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012. (Cité en pages x, 2, 35, 37 et 38.)

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992. (Cité en page 37.)

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012. (Cité en page 28.)

Xiaohui Shen and Ying Wu. A unified approach to salient object detection via low rank matrix recovery. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 853–860. IEEE, 2012. (Cité en page 55.)

Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993. (Cité en page 21.)

Guido F Smits and Elizabeth M Jordaan. Improved svm regression using mixtures of kernels. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2785–2790. IEEE, 2002. (Cité en page 30.)

Mark Steedman. Formalizing affordance. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 24, 2002a. (Cité en pages xv et 97.)

Mark Steedman. Plans, affordances, and combinatory grammar. *Linguistics and Philosophy*, 25(5-6):723–753, 2002b. (Cité en pages xv et 97.)

Thomas A Stoffregen. Affordances as properties of the animal-environment system. *Ecological psychology*, 15(2):115–134, 2003. (Cité en page 97.)

Alexander Stoytchev. Some basic principles of developmental robotics. *IEEE Transactions on Autonomous Mental Development*, 1(2):122–130, 2009. (Cité en pages xiii et 7.)

Ioan A. Şucan and Sachin Chitta. "moveit!", 2018. URL http://moveit.ros.org. (Cité en page 59.)

Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi: 10.1109/MRA.2012.2205651. http://ompl.kavrakilab.org. (Cité en page 59.)

Richard S Sutton. Verification, the key to ai. *on-line essay.[Online]. Available: http://www. cs. ualberta. ca/sutton/IncIdeas/KeytoAI. html*, 2001. (Cité en pages xiii et 7.)

Weimin Tan and Bo Yan. Salient object detection via multiple saliency weights. *Multimedia Tools and Applications*, (February), 2017. ISSN 1380-7501. doi: 10.1007/s11042-017-4725-7. URL http://link.springer.com/10.1007/s11042-017-4725-7. (Cité en page 54.)

David MJ Tax and Pavel Laskov. Online svm learning: from classification to data description and back. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, pages 499–508. IEEE, 2003. (Cité en pages 27 et 29.)

Constantine J Tsikos and Ruzena K Bajcsy. Segmentation via manipulation. *Technical Reports (CIS)*, page 694, 1988. (Cité en page 49.)

Michael T Turvey. Affordances and prospective control: An outline of the ontology. *Ecological psychology*, 4(3):173–187, 1992. (Cité en page 97.)

Tinne Tuytelaars, Krystian Mikolajczyk, et al. Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280, 2008. (Cité en page 21.)

Aleš Ude, Damir Omrčen, and Gordon Cheng. Making object learning and recognition an active process. *International Journal of Humanoid Robotics*, 5(02): 267–286, 2008. (Cité en page 51.)

Naonori Ueda, Ryohei Nakano, Zoubin Ghahramani, and Geoffrey E Hinton. Smem algorithm for mixture models. *Neural computation*, 12(9):2109–2128, 2000. (Cité en page 34.)

Emre Ŭgur, Mehmet R Dogar, Maya Cakmak, and Erol Sahin. Curiosity-driven learning of traversability affordance on a mobile robot. In *2007 IEEE 6th International Conference on Development and Learning*, pages 13–18. IEEE, 2007. (Cité en pages 12, 101 et 103.)

Herke van Hoof, Oliver Kroemer, and Jan Peters. Probabilistic Segmentation and Targeted Exploration of Objects in Cluttered Environments. *IEEE Transactions on Robotics*, pages 1–12, 2014. ISSN 15523098. doi: 10.1109/TRO.2014.2334912. (Cité en pages 49, 51, 118 et 120.)

Karthik Mahesh Varadarajan and Markus Vincze. Afrob: The affordance network ontology for robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1343–1350. IEEE, 2012. (Cité en pages 101 et 102.)

Alonso H Vera and Herbert A Simon. Situated action: A symbolic interpretation. *Cognitive science*, 17(1):7–48, 1993. (Cité en page 97.)

Nikos Vlassis and Aristidis Likas. A greedy em algorithm for gaussian mixture learning. *Neural processing letters*, 15(1):77–87, 2002. (Cité en page 32.)

William H Warren. Perceiving affordances: Visual guidance of stair climbing. *Journal of experimental psychology: Human perception and performance*, 10(5):683, 1984. (Cité en page 97.)

William H Warren Jr and Suzanne Whang. Visual guidance of walking through apertures: body-scaled information for affordances. *Journal of experimental psychology: human perception and performance*, 13(3):371, 1987. (Cité en page 97.)

Markus A Wenzel, Jan-Eike Golenia, and Benjamin Blankertz. Classification of eye fixation related potentials for variable stimulus saliency. *Frontiers in neuroscience*, 10:23, 2016. (Cité en page 54.)

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. (Cité en page 120.)

Kai Xu, Hui Huang, Yifei Shi, Hao Li, Pinxin Long, Jianong Caichen, Wei Sun, and Baoquan Chen. Autoscanning for coupled scene reconstruction and proactive object analysis. *ACM Transactions on Graphics (TOG)*, 34(6):177, 2015. (Cité en page 51.)

Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013. (Cité en page 54.)

Xinge You, Ruxin Wang, and Dacheng Tao. Diverse expected gradient active learning for relative attributes. *IEEE Transactions on Image Processing*, 23(7):3203–3217, 2014. (Cité en page 38.)

Philipp Zech, Simon Haller, Safoura Rezapour Lakani, Barry Ridge, Emre Ugur, and Justus Piater. Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271, 2017. (Cité en pages 12, 98, 99 et 101.)

Zhihua Zhang, Chibiao Chen, Jian Sun, and Kap Luk Chan. Em algorithms for gaussian mixtures with split-and-merge operation. *Pattern recognition*, 36(9): 1973–1983, 2003. (Cité en pages 33 et 34.)

Zhihua Zhang, Kap Luk Chan, Yiming Wu, and Chibiao Chen. Learning a multivariate gaussian mixture model with the reversible jump mcmc algorithm. *Statistics and Computing*, 14(4):343–355, 2004. (Cité en page 33.)

Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2814–2821, 2014. (Cité en pages 54 et 55.)

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3, 2003. (Cité en page 38.)

Wenbin Zou, Kidiyo Kpalma, Zhi Liu, and Joseph Ronsin. Segmentation driven low-rank matrix recovery for saliency detection. In *24th British machine vision conference (BMVC)*, pages 1–13, 2013. (Cité en page 55.)

Appendix A

Singular Value Decomposition (SVD)

SVD theorem *(For simplification only the real numbers case is considered)*

For a matrix M of size $m * n$ with real numbers, there exists a factorization of M called *singular value decomposition* of the form :

$$M = UDV^T \tag{A.1}$$

Where U is an $m * m$ orthogonal matrix, V is an $n * n$ orthogonal matrix and D is a diagonal $m * n$ matrix with non-negative values. Orthogonal matrices verify the following property : $O^T O = OO^T = I$. Where I is the identity matrix.

The diagonal values of D are called *singular values* of M.

Pseudo-inverse and Pseudo-determinant

From the SVD theorem a *pseudo-inverse* of non-invertible matrices can be computed as followed :

$$M^+ = VD^+U^T \tag{A.2}$$

Where D^+ is the pseudo-inverse of D, which is obtained by replacing all non-zero diagonal values with its *inverse*.

The pseudo-inverse computed as above fits the *Moore-Penrose inverse* definition :

For a matrix A of size $m * n$ with real numbers, its pseudo-inverse is defined as a matrix A^+ of size $n * m$ and verifying the following properties :

$$\begin{gathered} AA^+A = A \\ A^+AA^+ = A^+ \\ (AA^+)^T = AA^+ \\ (A^+A)^T = A^+A \end{gathered} \tag{A.3}$$

If a matrix is invertible then its pseudo-inverse is equal to its inverse.

Moreover, from SVD of a non-invertible matrix, a *pseudo-determinant* can be computed from its singular values. If this matrix is *positive semi-definite* then its singular values coincide with its eigenvalues and its pseudo-determinant can be obtained by multiplying its non-zero singular values.

A positive semi-definite matrix M verifies the following property :

$$x^T M x \geq 0 \quad \forall x \in \Re^n \diagdown 0 \tag{A.4}$$