Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue par :
Guido Manfredi

le vendredi 18 septembre 2015

Titre :

Learning Objects Model and Context
for Recognition and Localisation

École doctorale et discipline ou spécialité :
EDSYS : Robotique 4200046

Unité de recherche :
Laboratoire d'Analyse et d'Architecture des Systemes | UMR 7503

Directeur/trice(s) de Thèse :

Michel Devy
Daniel Sidobre

Jury :

Rapporteurs :
Helder J. Araujo, Universidade de Coimbra
Marten Bjorkman, Kungliga Tekniska Hogskolan

Examinateurs :
Eric Marchand, IRISA-INRIA
Frederic Lerasle, LAAS-CNRS

Mis en page avec la classe thesul.

Remerciements

First and foremost I would like to express my deep gratitude to my advisor Pr. Devy for his unconditional support, wise advices, enlighten knowledges, his patience, motivation and dedication in pushing me forward through the hardest moments toward the best outcomes, without him nothing of this would have been possible. I would also like to show my gratitude to my co-advisor Pr. Sidobre for the pearls of wisdom he shared with me, in robotics and other matters, for his exaltation and commitment to every project we went through.

My sincere thanks go to Pr. Danes for his kindness, to Pr. Lerasle for his sense of humour and his humanity and to Dr. Vandeportaele for the stimulating discussions. I express my warm thanks to Dr. Suarez-Roos for showing me that even the most difficult projects can be carried to their goals thanks to realism, calm and hard work. I would like to give a special thanks to Pr. Kragic, for allowing me to discover her lab and the Swedish culture, and to Dr. Karagiannidis and Dr. Bekiroglu for the interesting exchanges.

I am grateful for the funding sources : the Region Midi Pyrenées (CAAMVIS), the Agence National de la Recherche (ICARO) and the European Union (TWOEARS), along with all the partners involved in these projects.

My eternal gratitude goes to my beloved parents for their flawless support through all my life. I would like to express my love for my small Quokka whose everlasting smile lightened my sky when it was cloudy and warmed my heart when it was cold. Finally, I would like to thank my legs for always supporting me, my arms for always staying by my side and my fingers, I can always count on them.

I dedicate this thesis to my Princess,
my Sun,
my Heart.

Contents

List of Figures

Introduction

Dans un future proche, les robots serons présents dans nos foyers. Il prendront en charge des tâches de la vie de tous les jours tels que nous amener divers objets (nourriture, boisson, autres) et actionner les différentes machines qui peuplent déjà nos logements. Cela demande du robot une connaissance des objets présents dans le monde, une capacité à les identifier et manipuler.

Déjà aujourd'hui des robots sont introduit dans les chaînes de productions industrielles pour soulager les opérateurs des tâches les plus pénibles. Ils prennent en charge les parties ou un effort intense ou un effort de longue durée sont nécessaires. Peu à peu, ils participent à l'assemblage de pièces et à la vérification du travaille. Dans ce cas là, le robot doit être capable de voir les pièces qu'il manipule et de comprendre les différentes étapes d'assemblage. Encore une fois cela demande une connaissance des nombreux objets industriels et de leurs stades de construction.

Pour s'adapter rapidement au divers objets à traiter nous pensons que le robot doit avoir une base de données d'objet génériques pour lui permettre de gérer les situation les plus banales. Mais il doit également avoir la capacité d'apprendre de nouveau objets inconnus. C'est sur ce dernier points que se penche cette Thèse : la création de modèles perceptuels pour la reconnaissance et la localisation.

Des approches de modélisation existent depuis longtemps déjà et utilisaient du materiel dédié. De ce fait elles n'étaient pas pratiques d'utilisation. Plus tard avec le développement des méthodes de Structure par le Mouvement, la modélisation à été rendu plus accessible mais restait quelque chose de technique. L'arrivée des nouvelles camera RGB-D à bas coût à grandement facilité la modélisation d'objets, néanmoins elle produisent des modèles perceptuels complexes et de faible qualité due à la faible résolution des capteurs.

Nous pensons que la meilleur source d'information pour créer des modèles est l'utilisation d'images. En effet, il s'agit du média le plus répandu et le plus facile à acquérir avec des équipements de la vie de tous les jours. Nous nous intéressons donc dans ce travaille à la modélisation à partir d'images.

Les problématiques et solutions précédentes touchent la vision par ordinateur en générale. Dans le cas présent nous considérons la vision par un robot, cela nous permet d'avoir accès à des informations autres que celles présentes dans les images : le contexte du robot. Ce contexte peut se présenter sous une multitude de forme telles que : le lieux, les objets entourant le robot, la date, l'heure, la culture, etc. De manière générale on définit le contexte par rapport à un objet d'interet, et il est définit comme étant toute information autre que l'information visuelle fournie par l'objet.

L'apprentissage du contexte est un problème, chez l'humain elle demande des années d'expérience, mais un telle durée d'apprentissage n'est pas acceptable pour un robot qui doit être opérationnel rapidement. Nous pensons que l'apprentissage en minant le web est la solution, cela permet au robot d'apprendre le contexte rapidement et d'être à jour avec les nouvelles modes. Par ailleurs, une autre problématique important est la fusion des information visuelles avec les données contextuelles apprises.

Cette thèse propose des méthode de modélisation et d'apprentissage automatique du contexte pour rendre possible la reconnaissance et localisation d'objets à des fins de manipulation. Nous allons montrer que des images non contraintes suffisent pour former un modèle pour la localisation et que le contexte lieu et objets peut être appris automatiquement par l'expérience et depuis internet afin de faciliter la reconnaissance.

Ce travaille s'inscrit dans les problématiques de big data puisque nos méthodes permettent la modélisation d'objets et l'apprentissage du contexte depuis Internet.

Introduction

In a near future, personal service robots will have to handle daily chores like fetching objects, tidying up places, preparing meals and, in a more general way, using domestic appliances (Figure 1a). These tasks require knowing everyday life objects and interacting with them. However, when a robot is first introduced in a human environment, most objects are likely to be unknown. To handle the myriad of existing objects, service robots should focus on perceiving objects categories. These can be learned through autonomous learning or taught by a user. In this particular case, the user being no expert in robotics, the teaching method should be as natural, in a human sense, as possible.

Moreover, robots are being integrated at a growing pace in industrial applications. They provide a versatile workforce for repetitive or tough tasks (Figure 1b). In such situations, object's CAD models are likely to be known and the low number of objects to process allows for instance level object perception. For unknown objects, they may be taught to the robot by specialised operators using advanced techniques.

Additional concerns appeared ten years ago with the advent of collaborative robotics through the works from DLR about safe manipulators and the subsequent robots: KUKA LWR, ABB Frida, YASKAWA Motoman, etc. Future working cells will include physical interactions between robots and humans (Figure 1c); this implies joint human-object perception for better synchronisation. Again, this comes with challenges among which objects affordance or perception and reaction at a speed acceptable for humans.

In both the service and industrial robotics cases, the robot needs to quickly adapt to different tasks involving new objects. To handle objects perception, we believe that future robots will embed a database of known objects' perceptual models, from factory. These will be generic models, for example models of objects categories, to enable the robot to cope with common situations. However, as specific instances of these categories are met, specific perceptual models should be acquired. Additional models may be obtained by accessing a collaborative robotic web [1], by modelling directly visible objects or by creating new models from the human web. In this Thesis we put aside the categorisation problem to focus on instance level modelling with the two last solutions.

Note that in robotics the term model is often understood as CAD model because this type of model is used for recognition, localisation, grasping, obstacle avoidance and others robotic tasks. However, in the following the term model should be understood only as perceptual-model-for-recognition-and-localisation, not as CAD-model.

The first modelling approaches based on specialised hardware, like range finders or stereo rigs, required a specific setup, a large amount of calibration and expertise to be used. With the growth in computing power and the democratisation of digital images, the need to simplify modelling methods gave birth to Structure From Motion approaches [2]. These approaches allow modelling from a set of still images. No calibration is required, though some degree of expertise

(a) Service robotics (b) Industrial robotics (c) Industrial Cobotics

Figure 1: Three rapidly-developing applications of robotics.

is still needed when acquiring the images [3]. The resulting models allow computing 2-D textured descriptors for recognition and localisation [4].

The spread of new generation sensors providing RGB-D images has further improved modelling. Based on these sensors and efficient GPUs, it is possible to build precise models. The fact that the 3-D data is available on the fly induced a change in the descriptors as well. Nowadays, methods based on 3-D descriptors are privileged for recognition [5] and registration methods provide localisation [6]. These methods are also relevant for cases where a robot needs to model its environment for localisation purposes. For example, when it needs to localise itself inside a large object like a car or a plane. At this point, such approaches meet the Simultaneous Localisation And Mapping problematic.

The drawback of the previous methods is that they tend to produce complex models. Moreover, the object needs to be scanned with a sensor, this can be tedious and requires the object to be physically available in the first place.

On the other hand, the Internet used by humans overflows with data, especially visual data. On Facebook alone, more than 200,000 images are added every minute. Plus, advanced images such as panoramic or spherical pictures provide rich information of the depicted scenes. As the big data trend is currently gathering momentum, robotics will be able to rely on it in the future. We believe using the Internet as a source to create models, or to simplify the modelling process, is the way to go. Indeed, it is widely and quickly accessible and provides up-to-date information. Some works [7, 8] already explored this path by applying structure from motion methods to images obtained from large image databases such as Flickr. This is possible for scenes pictured by numerous people, such as historical monuments. However, to the best of our knowledge, no solution has been proposed for everyday objects.

So far, the problematics and solutions belong to general computer vision. Though, a robot is more than a sensor, it has a context, for example its location in space and time. When performing tasks involving a human, the context of the human should also be accounted for. This implies learning numerous knowledge associated with humans like human environments, tools or actions. As pointed out by Divvala et al. [9], there are many types of contexts: local pixel, scene gist, geometric, semantic, photogrammetric, illumination, weather, geographic, temporal, cultural, etc. In the following, we define the context in a general fashion as being all information that is not directly related to the visual information strictly needed by the task at hand.

Merging the contextual and visual information has been a main concern in the vision community. In [10], the authors show that these methods can be grouped in two groups: classification

(a) Place context (b) Objects context (c) Action context

Figure 2: Three types of context commonly used. 2a The cow is out of context in the sky. 2b The mouse, keyboard and monitor appear frequently together. 2c Though the straw gives almost no visual information, the drinking gesture allows identifying it easily.

and graphical methods. Most works use as context the environment [11], other objects [12] or human actions [13] (Figure 2). Usually no more than two contextual sources are used at the same time.

Though works relying on context are numerous, learning the context remains an open problem. Most works learn it from a limited dataset specifically designed for this task. Unlike these works, for practical cases, we advocate in favour of robots able to learn object's context autonomously. Similarly to the object's models, one can envision a robot being delivered with a database containing general contextual rules. However, there are many other rules that will depend on the owner's culture, geographical position, socio-economic status, etc. For these cases a robot should be able to learn autonomously from experience. The drawback of such approach is the time required for the robot to learn: it will have to encounter similar situations numerous times before being able to learn a contextual rule. To learn quickly, accessing the Internet and being able to extract contextual information from it seems crucial. This may become increasingly easier as the Internet of Things provides more and more data about the way humans live.

This Thesis's ambition is to propose concrete solutions to the aforementioned problems, solutions applicable beyond the framework of a research project. As will be seen shortly, this is especially reflected by our contributions about using the Internet as a learning source. An always present objective is building autonomous methods. For object modelling, we propose a structureless model and an associated localisation method which allow performing the modelling task with an image search on the web. For context learning, a method is proposed to learn the context from Amazon.com, yielding a co-occurrence matrix for a set of objects from the objects names only. Finally, for visual and contextual data fusion, we show that simple logic rules are sufficient to build a probabilistic graphical model thanks to Markov Logic Networks.

From a technical point of view, the contribution of this Thesis is twofold. First, an insight into the classical object modelling approaches and the resulting models shows that a trade-off between complexity and robustness can be reached. It is shown that such models are simpler than classical models and that their lessened robustness can be handled by a robot. With simple models in mind, the scanning process is not necessary and a method is proposed to model objects from objects measures and a small set of non overlapping images. To remove the need of the

object physical presence, a second modelling method relying exclusively on images is proposed.

The second contribution is a set of methods to learn and use contextual information to improve recognition. A method allowing a robot to autonomously learn place-object relationships is described and an inference algorithm merging visual descriptors and the place-object context is proposed. A similar work is done for the object-object contextual relationships. A learning method is proposed to obtain such relationships from the Internet and an inference method using this data is presented. Collaboration with another PhD student has been initiated on the topic of object-human contextual information, i.e. the relationships between an object and the gestures humans execute when using it, but time was short to make a significant contribution.

We start by defining the concept of modelling process and choosing a particular modelling process for rigid objects recognition and localisation. For this particular case, a review of the state of the art allows listing the main limitations of classical methods. Each limitation is addressed in turn to reach a modelling method free of these problems. With the modelling problem addressed, the rest of this Thesis focuses on solutions to handle numerous objects. As the main obstacle seems to be the recognition step complexity, we present methods to relieve the recognition problem through the use of context. Finally, an industrial application allows zooming out of the object perception task to better understand its role in a full robotic architecture. Each Chapter's content is detailed hereafter.

The problem of object modelling is presented in Chapter 1. The modelling process, in a general way, is split into four components: real system, data, properties and model. Our choice for each component is justified in the particular case of modelling for robotics. With a solid notion of what is modelling for manipulation in robotics, the different components of the modelling process are discussed. In the context of this Thesis, the focus is put on modelling for recognition and localisation using local textured descriptors from RGB and RGB-D images.

The actual modelling process is presented in detail in Chapter 2, along with a state of the art of modelling from images. The whole modelling pipeline is scrutinized, from the acquisition of data from the actual object to the creation of the model itself. The classical modelling methods are presented and existing solutions are compared. Two problems emerge from this study: the models complexity, which tends to get high, and the difficulty of building a model, which requires both time and expertise. These problems are dealt with in the three following Chapters.

Chapter 3 investigates the complexity problem. It presents two paradigms used when modelling: the multiple views and robust features paradigms, and proposes to find the correct balance between both paradigms. This is done with the idea of reducing the modelling difficulty in mind. Experiments are set up to show how a limited set of images from the object can be enough to build an acceptable model. Besides, this Chapter introduces the notion of acceptable blind spots and blind spots aware model. The acceptable blind spots are the quantity of blind spots a model can have while still allowing the robot to fulfil its tasks. The blind spots aware model is an object model with information about the position and size of its blind spots so the robot can plan accordingly. This Chapter is concluded by proposing a balance between number of images in a model and robustness of a descriptor.

To simplify modelling to the extreme, Chapter 4 proposes a method to model easily an object from a limited set of images and a shape a priori. Relying on texture back projection, this allows modelling an object with a priori data that can be gathered easily like the object's dimensions and its shape. An example is provided for planar, cuboid and cylindrical objects. Though easy to build, the models are approximations. To make sure this does not hamper the localisation, the precision of localisation with those models is estimated. Results show that the localisation

precision is sufficient for manipulation.

On the other hand, Chapter 5 does not propose a modelling method. It uses the simplest model possible: images, with no other information. This type of modelling not only allows modelling with images taken from the internet but also handles objects which have discontinuous textures. The contribution is a new localisation algorithm which allows localising objects from an image and a RGB-D input. Simulation results show that the precision is at least as good as Perspective-N-Point algorithms in the calibrated and uncalibrated case. An actual experiment is also performed in the calibrated case and localisation results are shown for two objects.

Previous Chapters handled the modelling problem and focused on simplifying the modelling process to the extreme. In a real situation, robots can encounter thousand of different objects and possibly tens at the same time. State of the art recognition algorithms are not designed to handle such mass of data, even less at high speed. In order to facilitate the recognition, we propose to use a source of data limited in a general computer vision case but rich in the robot case: the context. As stated earlier, a robot has many different contexts. In this work, we deal with two types of contexts: the place and the other objects.

Chapter 6 presents a method for a robot to autonomously explore, model, segment and organise a site. By using navigation and localisation capabilities the robot is able to build a 3-D map of its environment which is then processed to extract various places of interest and their relationships. This data allows linking objects to areas and computing probabilities about where objects are likely to appear. Then, a cascade based method is presented to merge simple visual cues and the objects likelihood to appear in given places. Results show that this allows simplifying the recognition problem by discarding unlikely candidates.

The second type of context, co-occurrent objects, is tackled in Chapter 7. A method to automatically extract co-occurrence data from Amazon.com is first presented. By using Amazon.com features, we propose three different types of matrices to describe objects co-occurrence. Then, a method to merge these matrices with visual data is proposed. Though Markov and Conditional Random Fields are the preferred method, we show that Markov Logic Networks are a more interesting choice which can be more easily used and has been shown to produce better results. The matrices extracted from Amazon.com combined with a state-of-the-art visual detector through a Markov Logic Network are set to be used on the COCO dataset, specifically designed to provide contextual information. Results are pending while the COCO dataset is finished.

The final Chapter zooms out of the vision problematic to show a full robotic application where the object recognition and localisation is but a part of the full architecture. This allows putting the previous work into perspective and showing that the problems answered in this Thesis are but a mere fraction of the complexity of a whole robotic system. This Chapter treats of a human-robot collaborative joint assembling task where a ball bearing is assembled by successive steps done by the human and the robot with some steps requiring both of them to work together. The architecture is presented in detail and lessons learned from this project are gathered.

Chapitre 1

Lorsqu'un robot observe une scene, elles contiennent deux parties : une partie statique, qui correspond aux murs, meubles, etc. ; et une partie dynamique qui correspond aux objets deplacables et manipulables. A un instant donné, la partie statique peut être localisée une fois pour toutes, par contre, les différentes parties dynamiques doivent être reconnues et localisées indépendamment. Pour ce faire, le un robot doit avoir des informations à priori sur ce qu'il cherche, un modèle des objets. Ce modèle est comparée à la scène visible pour reperer les objets d'interet.

Dans ce Chapitre nous nous penchons sur la façon de créer des modèles pour des objets d'interet. Nous définissons un processus de modélisation comme étant la combinaison de quatre éléments : un systeme reel à modéliser, les propriétés de ce systeme à obtenir, des capteurs permettant d'obtenir ces propriétés en observant le system reel et enfin un modèle permettant d'enregistrer les propriétés de manière compacte. Par la suite nous justifions notre choix pour chacun de ces éléments en commencant par les propriétés que nous souhaitons extraire de la scene pour chaque objet.

Puisque nous nous interessons au cas de la manipulation robotique, les propriétés à extraire de ce monde sont : l'identité des objets, leur pose et leur structure. L'identité de l'objet permet de le segmenter dans la scene et de le distinguer par rapport à d'autres objets. La pose de l'objet est sa position dans l'espace 3-D, elle permet d'intégrer l'objet dans la représentation 3-D que le robot possède du monde. Finalement, la structure de l'objet représente sa forme dans l'espace 3-D, elle est utile pour calculer des trajectoires d'évitement ou au contraire pour trouver des points de saisie sur l'objet.

Le systeme réel est, dans le cas présent, le monde réel que l'on partage avec les robots. Ce monde réel impose un certain nombre de contraintes qu'il faudra prendre en compte au moment du choix du modèle. Ces contraintes sont : la pose des objets qui peut changer leur apparence, la luminosité, l'occlusion, la multiplicité des objets de même apparence, la vitesse de traitement requise en milieu humain et la precision necessaire pour que le robot puisse manipuler les objets.

Afin de percevoir au mieux le monde et d'obtenir les trois propriétés d'interet (identité, pose, structure), nous choisisons de travailler avec des capteurs présent sur tous les robots et accessible à faible cout : des cameras RGB et RGB-D.

Finalement le choix du modèle dépend fortement des contraintes du system réel. En effet, le modèle doit contenir suffisament d'information pour prendre en compte les variabilités apportées par chacune de ces contraintes. Les information visuelles suceptibles d'etre incluse dans le modele sont : la couleur, les contours 2-D, contours 3-D, la texture ou un mélange des précédents. Par ailleurs ces informations visuelles doivent être extraites à une échelle donnée : pixels, voisinage de pixels, super-pixels, region ou image.

Afin de trouver le modèle le plus adapté à notre situation d'un robot évoluant dans un environement humain nous comparons plusieurs descripteurs en fonction des contrainte du monde determinées plus haut. Il ressort de cette comparaison que les descripteurs locaux de texture offrent le meilleur compromis pour notre situation.

Le reste de ce travaille se concentre donc sur la modélisation d'objet à partir de déscripteurs locaux de texture pour la reconnaissance et la localisation.

1

Modelling for Object Manipulation

> I think that reality exists and that it's knowable.
>
> Jimmy Wales

Knowing the reality is an important requirement for a robot to perceive the world. Getting data through its sensors, a robot needs to know what it is looking for to make out the different parts of the world. This a priori knowledge is provided to the robot in the form of models.

In the case of an indoor environments, the world can be separated into static parts (heavy furnitures, walls, etc.) and dynamic or moveable parts (chairs, small objects, living beings, etc.). Static parts can be segmented and learned once and for all. This is not possible for the dynamic and moveable objects, they need to be recognised and localised at each instant.

To summarise, dynamic objects need to be recognised and localised, which is done thanks to known models. The following goes over the problem of setting up a pipeline to build these models. Each part of the pipeline is analysed and a specific choice is done for each part.

1.1 Modelling in the wild

1.1.1 Main components

Before going into the details of object modelling methods, let's look at modelling from a general perspective. We take a look at the different components of a modelling process, and show how they are chosen when building a modelling system for robotics.

We define a modelling process as four components: the real system, the available data, the properties to mimic from the real system and the model itself.

- **Real system**: is the phenomenon to represent in the model. For example, in the robot case, the world around it is the real system, with its shapes, colors, weights, speeds, etc. Often, the real system is not accessible directly and sensors are used to grab data about this reality.
- **Data**: The properties of the real system are only accessible through sensors. Each sensor gives partial information about one or multiple properties. The whole information provided by these sensors is the available data. In the example of a robot looking at a scene, the real system is the scene. The digital signal provided by the camera is the data.

Table 1.1: A few examples of modeling processes and their components

Example	Real system	Data	Properties	Model
World (by human)	World (physics sense)	The five senses	Shape, color, taste, sound, etc.	Brain model (unknown)
Music	Sound waves ordering	Ear signal events ordering	Frequency, Harmony, Rhythm, etc.	Music Sheet
Time	Events ordering	Sens of events ordering	Ordering, Duration	Time as number
World (by robot)	World	RGBD images, laser scans	Localisation, appearance	SLAM maps, Object models

- **System properties**: The aim of a model is to mimic a set of properties from the real system. These properties are the information of interest to be reproduced. However, properties can be reproduced only if there is data about them, so the properties to mimic are limited by the available data. For example, a robot sensing the world through a camera can't tell the object's weight, because the weight property is not available in images. Moreover, the more properties reproduced in the model, the more complex the model. When fast processing is required, one chooses the minimum set of properties necessary for the task at hand.

- **Model**: A model is made of one or various mathematical spaces. It is a simplification of the data space which retains the properties to mimic from the real system. The choice of these spaces impacts the robustness and precision of the predictions made with the model. For example, on an image, luminance patches are simple models but they only match the data they come from. On the other hand, interest points handle some data variability but they are expressed in complex mathematical spaces. In the robot example, the model can be a visual SLAM map of the environment, i.e. a set of geometrical features expressed in a reference frame in 3-D space and a set of associated visual descriptors.

When designing a modelling process for robustness, there is a trade-off between the quantities of data needed and the complexity of the models. Indeed, for an equal robustness to changes, simple models require more data than complex ones.

1.1.2 How to: Modelling in Robotics

When elaborating a modelling task, the properties of interest that the model has to mimic from the real system are the first thing to consider, they are chosen once and for all. Then the real system should be analysed to reveal the constraints and difficulties that appear when trying to get such properties. Next comes the selection of sensors, they should provide relevant data to reproduce the desired properties. Usually, only a subset of the raw data is used. After defining the data, mathematical spaces capable of reflecting the properties of interest are selected. Finally, one chooses the modelling process, a function going from the data space to the model space which preserves the properties. This function allows creating new models.

In the next sections, this reasoning is applied to the specific case of object perception for manipulation. More details are provided for each part of the modelling process and our choices for the modelling components is explained.

1.2 Properties for Manipulation

When asked to manipulate an object the robot is confronted with four challenges: recognising the object of interest, localising it, grasping it and avoiding obstacles while manipulating it. The grasping and obstacle avoidance task used to need information from the model, typically some shape information. However recent works and 3-D sensing capabilities [14] allow grasping without a model. Obstacle avoidance can also be done from a map built online from the output point cloud of a 3-D sensor [15]. For these reasons this work focuses on the recognition and localisation problems.

1.2.1 Recognition Problem

Robots must be able to identify objects, and possibly classify them, in order to retrieve the correct localisation model from the models database. The solution to this problem answers the questions: which object is this? To which group of objects does it belong? This is usually done by matching sensory cues with available models. However, it implies problems such as model invariance or cues density and uncertainty. Moreover, in a robotic framework, sensors can be moved so the recognition process can be improved by active action from the robot (see [16, 17]).

1.2.2 Localisation Problem

When the object is identified, in order to be manipulated, it should be localised in space. To grab it, the robot needs to estimate the object position with respect to itself. The localisation process relies on the cues position in sensor space and their position in object space to compute the relative motion between the object and the sensor. Usually, recognition finds the correct object model and localisation uses this model to get the object pose.

1.2.3 Structure Problem

The structure refers to 3-D information about the object's shape. It is crucial for tasks such as motion planning, collision avoidance [18] and online grasp planning [19]. As noted at the beginning of this section, saving the structure is not mandatory as RGB-D sensors can provide it on the fly. However, including it in the model can facilitate the localisation problem.

Having defined these properties, the following takes a look at the real world to find the constraints which could prevent obtaining these properties.

1.3 Real World and Constraints

This Thesis is limited to indoor scenarios. In such scenarios, constraints are considered from the object recognition and localisation point of view.

- **Pose**: Objects in a human environment can be placed in any position. Moreover, their apparent scale will change according to the robot-object distance. The model should be robust to point of view related transforms.

- **Light**: Depending on the season, weather and time of the day, the light in a room can vary greatly in intensity and color range. Objects appearance will vary accordingly. The model should be able to handle such lighting changes.

- **Occlusion**: Other objects can partly hide the object of interest. It is likely, for an object held in a human hand, to be occluded by the hand. Some convoluted objects may occlude themselves. In any case, the model should deal with partial views.
- **Multiplicity**: Identical objects can appear multiple times in a same scene or different objects but with similar appearances can be present. The model should discriminate such objects.
- **Speed**: In a human context, robots must act with a human acceptable speed. In the same way a computer taking too long to display a web page is annoying, a robot taking too long to act can be irritating. Thus, the model shouldn't take too long to be created or to be used.
- **Precision**: Whether for collision avoidance, grasping or many other tasks a robot performs, there is always a safety padding aimed at handling imprecision. So a robot has some error margin. Nevertheless, the model should provide a localisation precise enough for objects manipulation.

These constraints need to be considered when choosing which data to use and in which mathematical space the resulting model is expressed. After analysing the constraints from the real world, the next step is the sensors choice.

1.4 Data and Sensors

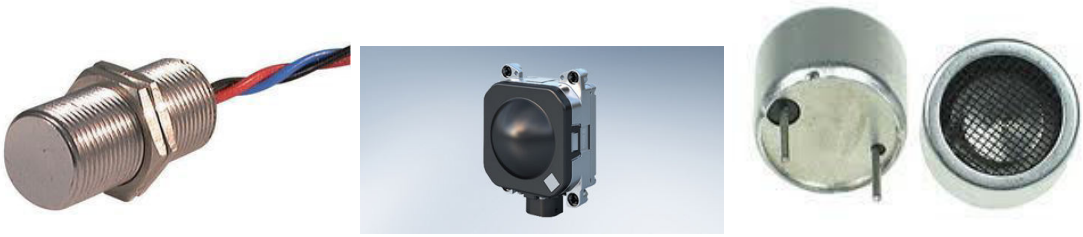
In robotics, different kinds of sensors are available: color, depth, laser, ultrasonic, radar, magnetic, etc. (cf. Figure 1.1). However, due the high variability in objects material, many of these sensors are not relevant for generic objects perception. Because they are the most common, almost mandatory for indoor robots, we focus on two types of sensors : RGB and RGB-D cameras. They received a lot of attention from research, so there is a great choice of algorithms and software for manipulating their output. We choose not to work on videos for complexity reasons. A robot has many processes running simultaneously, processing whole videos would monopolize too much processing power. Instead, we focus on object detection, recognition and localisation, using still images regularly sampled from the camera's stream. These are the preliminary steps to various video processing that can be done at a later stage, when the robot focuses on a single task, e.g. visual servoing when grasping an object. Nevertheless, updating the objects-robot position using features tracking in a video is out of the scope of this work.

Even considering still images, a camera's raw output contains too much information to process directly. Usually, only parts of the raw data are used, these parts are called descriptors.

1.5 Descriptor Spaces

When working with visual data, the model space defined earlier is usually called descriptor space. Descriptors are pieces of information extracted from an image, they are specifically designed to retain a given type of information. Each type of descriptor has strengths and weaknesses which make it adapted to some problems but not for others. Descriptors use different pieces of information from the raw image, sometimes mixing them. The pieces of information are called visual cues. The visual cues can be computed at various scales on the image. This will impact their complexity, robustness and spatial resolution.

(a) RGB sensor (b) Depth sensor (c) Laser sensor



(d) Magnetic sensor (e) Radar sensor (f) Ultrasonic sensor

Figure 1.1: Pictures of different sensors used in robotics.

1.5.1 Visual Cues

Visual cues can be of different types: color, gradients, shape, mathematical functions (e.g. wavelets), etc. Some classical cues have emerged in the literature, they are described in the following.

Color: Color cues are very rich cues due to the large amount of information they carry. Their main drawback results from the poor color handling from visual sensors. Color constancy with respect to illumination color is an open problem, information on the nature and the way light is projected on the scene is needed to extract the color properties. Thus color observed from a camera can be very different when observed by a different camera. Many color spaces have been developed [20] to lower this effect but this is far from solved. Another problem with color method is the complexity due to the large amount of data.

Contours 2-D: In the eighties, a large amount of works where devoted to contour based recognition and localisation methods independent from the object shape. However, their sensibility to lighting conditions makes them hard to use, even indoors with controlled lighting.

Contours 3-D: The recent availability of cheap 3-D sensors made the 3-D contour cues popular. Many methods have been proposed [5], often extensions of 2-D descriptors. There is still a lack of consensus about 3-D cues quality and the additional dimensions make them slow to compute.

Texture: The development of the SIFT [21] feature made the textured cues attractive. They are fast to process, can provide enough information for high robustness. A lot of work has been done on textured cues for scenes, objects, human recognition, classification, localisation, etc.

Mixed: When enough processing power is available, methods using various cues at the same time can be used. The main problem being that in order to take full advantage of this kind of cues, the considered objects must have all the "sub-cues". For example, for a color/texture/3-D

shape cue, the considered object must have an outstanding 3-D shape, be textured and coloured. Mixed cues are used in [22] for increased robustness.

1.5.2 Scale

When computing a descriptor, one must choose how much information it retains from the image. The bigger the described area, the higher the discriminative power, but also the lower the generalization. Descriptors can describe from a pixel to a whole image.

Pixels: Some descriptors describe a pixel using other pixels around it. For example in [23], the descriptor is computed from differences of pixels intensity.

Local: Local means that a small neighbourhood around a given position on the image is described. This is the case for the many Local Texture Features presented in the literature [24] [25] [26].

Super pixels: Super pixels are small homogeneous parts of the image, computed with a super pixel segmentation algorithm. In [27], the authors describe super pixels with a mixed set of descriptors including color, laws masks, oriented edges, etc.

Region: A region, contrary to super pixels, has a semantic meaning. It can be part of a landscape, objects, etc. Regions are described in [28] thanks to various texture kernels and 3-D shape.

Image: A whole image contains a lot of information but coding all of it into a single descriptor is a hard work. Still, the authors of [29] managed to reduce all this information to a few dozen bytes using fisher kernels.

Depending on the problem at hand, the most informative scale should be used.

1.5.3 Comparisons

In the following, examples of visual cues at different scales from the literature are provided. The strength and weaknesses of the resulting descriptors are stressed.

Because of their current popularity, this comparison starts with the 3-D descriptors. Historically, the spin-image [30] is one of the earliest 3-D descriptors. For a given point, it computes in cylindrical coordinates an histogram of the positions of neighbouring points. The work from Tombari et al. [31] introduces a descriptor built from merging histograms of normals differences and local geometrical information. The authors of [32] propose the use of a fast 3-D histogram descriptor for object recognition. This feature is local and describes 3-D texture around a point. More descriptors and a thorough comparison can be found in [5]. Though precise, robust to occlusion and pose change, these approaches suffer from slow processing time and confusion when presented with identical objects. Moreover, many everyday objects have similar shapes, so they cannot be discriminated by such descriptors alone. Finally, objects with highly symmetrical shapes, e.g. cylinders, provide few 3-D features.

The authors of [33] generalise a haar wavelet descriptor to a volumetric spatio-temporal descriptor for event detection in videos. This descriptor is very fast, however it is known to have poor occlusion resilience and can't handle objects pose change. Moreover it only works for videos.

In [34], the authors elaborate a RGB image descriptor using color histograms of gradients for image matching. This method extends the local texture descriptor SIFT [21] to color. Though adding more discriminative power, the use of color increases complexity and makes the descriptor subject to the camera's color errors. A model created with some camera may not work with another one.

Table 1.2: Classical descriptors from the literature compared on the criteria of pose, occlusion, multiplicity, speed and precision. The comparison is done empirically based on the results provided in each work.

work	cue	scale	pose	light	occlusion	multiplicity	speed	precision
[33]	3-D box	multi-scale	+	++	+	+++	+++	++
[32]	3-D texture	local	+++	++	++	+	+	++
[34]	2-D color	local texture	+++	+	+++	+	++	++
[35]	contours	local	+++	+	+	++	+++	++
[36]	texture	global	++	+++	+++	+++	+	++
[22]	mixed	mixed	+++	++	+++	++	+	++
[37]	texture	local	+++	+++	+++	+	+++	+++

The descriptor elaborated in [35] describes objects with a 3-D shape template matching approach. This provides speed, precision and robustness to pose. However, it struggles to handle occlusion.

The authors of [36] use a global descriptor for object recognition and classification. These features handle quite well occlusions and multiplicity. They lack robustness for pose estimation and their complexity makes them slow.

In [37], the authors extend the SIFT descriptor to make it robust to affine transformations. This descriptor ends up robust to pose change, occlusion while retaining high speed and precision.

Finally, in [22], the authors mix color, 3-D shape and 2-D texture descriptors to obtain a robust algorithm. Though handling pose changes, occlusions and multiplicity, the overly complex model renders the process slow. Moreover, the lack of consensus on the cues can lead to a lower precision in some cases.

Table 1.2 offers a comparison of the most successful descriptors from the literature. They are compared on the constraints listed earlier, specific to the object robotic manipulation problem.

For a more complete list see [38] for color, [39] for shapes, [4] for texture and [5] for 3-D.

From the comparison, it appears that the best suited descriptors for the problem at hand are the local texture descriptors. They are robust to pose changes, handle occlusion, have limited sensitivity to lighting changes, have high speed and precision. Moreover, though they don't handle naturally multiple identical objects, methods have been proposed to address this limitation[40] [41].

The next section provides more arguments to explain our interest towards textured objects.

1.5.4 Local Texture Descriptors

The majority of grocery products are textured objects due to their packaging. Alone they represented 22,16% of the average EU household expenditure for the 2010 year [42]. This imply that textured objects are highly represented in every day manipulated objects. Thus the necessity of perceiving them for a robot. Moreover, the fact that new ones are bought regularly also imply that textured products change frequently. For example, new products can appear, a same product can change packaging, etc. Thus a strong need in modelling capabilities to be able to handle quickly and painlessly new objects.

Most textured approaches are based on interest points. Other well established methods also use interest points, for example SLAM [43], gesture recognition [44], etc. Using the same source of information allows pooling resources and saving computer power.

Textured approaches, more specifically those based on interest points and 3-D models, allow using localisation methods both precise and robust to various kind of noises. According to [45], these are the more precise localisation methods.

Because of their high representation in everyday life, their tendency to be manipulable objects and the need for creating new models regularly, the rest of this Thesis focuses on textured objects.

1.6 Conclusion

This chapter introduced the modelling problem and its components: the real world, the properties of the system, the data and the model. It described our approach when facing a modelling problem by considering in turn each of the four components. Then it is shown that for robotics manipulation, the model should contain three properties : identity, pose and structure. These properties imply a set of constraints in the real world. After choosing a type of sensor, a comparison of various descriptor spaces showed that local textured descriptors are the best answers to the given constraints. Moreover, this Chapter showed that, for various reasons, textured objects are good candidates for new modelling methods. Thus, the rest of this work focuses on building perceptual models for recognition and localisation using local texture descriptors. The next Chapter introduces the standard pipeline used for modelling objects. Existing tools are implemented, evaluated and their limitations are raised.

Chapitre 2

Dans le Chapitre precedent, nous avons definis comme objectif la modélisation pour la reconnaissance et localisation d'objet à partir de descripteurs locaux de texture. Dans le présent Chapitre nous allons présenter les méthodes existantes, analyser leur limitations.

Nous commencons par une revue des methodes de reconnaissance et localisation existantes. En commençant par le methodes d'asservissement visuel qui peuvent être également utilisées pour faire de la localisation d'objet (ViSP), puis les méthodes basées sur des descripteurs 2-D (MOPED). Nous poursuivons avec l'approche ROS utilisant une methode ICP ou des boites englobantes selon que l'objet est connu ou non. Puis nous passons au méthodes à base de descripteurs 3-D, dont un grand nombre sont implémentés dans la bibliothèque PCL. Nous finissons en parlant de l'approche mixte descripteurs 2-D et 3-D en illustrant avec la méthode Linemod.

Bien ces méthodes aient fournis de bon résultats, leur utilisation ne s'est pas popularisée. Nous pensons que le problème vient de l'existence d'un nombre limité de models perceptuels d'objets et de la difficulté à produire de nouveau models perceptuels. Pour ces raisons nous nous concentrons dans la suite sur les methodes standard de modélisation d'objets pour la reconnaissance et la localisation.

Afin de créer des models perceptuels qui prennent en compte les différents aspects que peut prendre un objet en fonction du point de vue d'observation, la methode standard est de prendre de très nombreuses vues de l'objet sous des points de vue differents et de les inclure dans le model. La génération de ces vues peut se faire en ligne, en scannant l'objet avec un capteur, ou virtuellement à partir d'un model CAO de l'objet, en scannant l'objet grâce à un capteur virtuel.

La méthode en ligne possède l'inconvenient d'être manuelle, sensible au bruit introduit lors des acquisitions et de produire un modèle specifique aux conditions présentes au moment de l'acquisition. Au contraire la méthode virtuelle est automatique, permet de ne considérer que l'objet et de simuler un grand nombre de conditions différentes. Par ailleurs, les méthodes virtuelles permettent de générer un nouveau modele du même objet si le besoin apparaît. Le principale inconvénient vient du fait qu'elle sont dépendantes de la qualité du modèle CAO utilisé. Néanmoins, leur practicité en fait la méthode de choix pour la robotique et nous choisisons l'approache virtuelle pour générer des vues d'un objet.

Avec ces méthodes vient le problème d'obtenir un modèle CAO des objets que nous souhaitons modéliser. Les sources sont diverse, la plus riche est constituée de bases de données présentes sur internet, les modèles destinés aux designe/architecture/mondes virtuels sont souvent peu précis. A l'inverse, des bases de données créer par des communautés de chercheurs possèdent des modèles précis mais rapidement obsolètes car rarement mis à jour.

Plutôt que de prendre dans une source existante, nous pensons que le robot doit être capable de créer ses propres models. Pour cela trois grandes familles d'approches existent : les méthodes type Structure par le Mouvement (SM), les méthodes type SLAM et enfin celles basées sur des capteurs de profondeur. Afin de comparer les performances des méthodes SM et avec capteur de profondeur, nous comparons les logiciels 123DCatch (SM) et ReconstructMe (profondeur) sur un jeu de quatre objets au formes differentes présentant tous des zones texturées. Les résultats montrent que les méthodes SM souffrent d'un manque de précision en terme de forme alors que la texture est reproduite de manière fidèle. A l'inverse, les méthodes profondeur reproduisent la forme avec precision mais la texture est de faible qualité. Selon l'application, l'une ou l'autre des méthodes peut être choisie.

Une fois le modèle CAD obtenu, celui ci doit être échantilloné pour obtenir des vues réparties de manière uniforme autour de l'objet et à des distances différentes. Trois méthodes

d'échantillonage sont proposées : electrons, où les vues sont choisies de manière à minimiser une force electromagnetique entre les positions de camera ; simple spiral, où les vue sont prises à interval régulier sur une spirale normale ; spirale dorée, où les vues sont prises à interval régulier sur une spirale dorée. Cette dernière méthode est privilégiée car elle permet une couverture plus uniforme des vues possibles.

Avec la pipeline de méthodes décrite ci-dessus, il est possible de créer un modèle pour une approche de reconnaissance et localisation. Afin de tester la pipeline, nous créons plusieurs modèles pour la méthode Linemode, en changeant les paramètres d'échantillonage. Il ressort de cette expérience que cette pipeline standard produit des modèles contenant de nombreuses vues (plusieurs centaines), de grande taille (plusieurs centaines de Mo), qui demandent un temps de chargement en mémoire long (plusieurs secondes) mais dont le temps de traitement augmente avec la taille (de 17 à 20ms).

Ce Chapitre se conclut en pointant du doigt deux grand défaut de la pipeline précédent. Premièrement, elle est compliquée à mettre en place au niveau logiciel et sont utilisation demande une expertise de la part de l'utilisateur, que ce soit pour obtenir le model CAD ou pour choisir convenablement l'echantillonage. Le deuxième défaut de ce type d'approches est qu'elle produit des models possèdant trop de vues et donc trop complexes. Les trois Chapitres suivants proposent de solution à chacun de ces deux problèmes.

2

Modelling Methods

In the previous Chapter, we have set our goal to modelling for object identification (name) and localisation (pose in 3-D space) possibly relying on its structure (shape in 3-D space). For clarity, in the following, the term *model* is used to designate the perceptual model needed for recognition and localisation while the term *textured mesh* refers to a CAD model with associated texture.

The following goes over popular methods for recognition and localisation and shows that they all rely on a model. Two modelling methods to create perceptual models are presented: online and virtual. We show that, for practical reasons, the virtual approach based on textured meshes is preferable. Then, we provide an overview of the principles used to obtain textured meshes and a thorough review of the state-of-the-art in building textured meshes. A comparison of two available methods follows. The model creation pipeline is completed by showing three techniques used to generate views from a textured mesh. Finally, two drawbacks of this modelling pipeline are put forward, namely the fact that they are too complicated to set up and use, and that they yield models more complex than necessary.

2.1 Existing Methods for Recognition and Localisation

Classically, objects identity and localisation are computed through a detection, recognition and pose estimation pipeline. We define detection as finding one or various regions of the scene belonging to objects of interest. In the recognition step, the detected regions are compared through descriptors with a set of known models to find the corresponding models. By comparing the object appearance from the scene and its structure from the model, pose estimation finds the 3-D transform from the camera frame to a frame associated with the object and defined in the model. Thus, a model needs to include visual descriptors, for recognition, plus a structure with an associated frame, for localisation. Currently, various methods are readily available to perform recognition and localisation, they rely on different hypothesis and visual cues.

As shown by Comporte et al. in [46], pose estimation can be defined as the dual of visual servoing. A virtual camera is moved to minimize the error between the back-projection of a known 3-D mesh and the real scene. The real camera pose is the one minimising the error. Though detection and recognition must be taken care beforehand, e.g. with textured descriptors, these methods allow pose estimation with many kinds of features through the use of an interaction matrix [47], i.e. a matrix which encodes the features as constraints for a control law. Such approaches can be realised with ViSP [48], a framework which allows to effortlessly implement visual servoing solutions [49].

For textured objects, detection and recognition can be performed by matching visual 2-D descriptors from the scene with descriptors extracted from known models. Object recognition has been intensively studied, particularly in the Content Based Image Retrieval (CBIR) community [50]. Strongly motivated by challenges such as the Pascal-VOC challenge [51], highly efficient algorithms have been developed [50, 52]. Independently, impressive results have been presented, for example the work of Krizhevsky et al. in [53]. In the robotics community, various works have tackled the specific problems of recognition for robotics: clutter [54], confusion [40], complexity [22]. From an implementation point of view, libraries such as OpenCV [55] and VLFeat [56] provide tutorials and code to quickly build detection and recognition applications.

The pose estimation part is usually a perspective-N-point problem. A model containing 3-D points from the object geometry is compared with 2-D features acquired from a camera, then projective geometry relationships allow retrieving the object pose with respect to the camera. This subject has been extensively described in the central work of Hartley and Zisserman [57] and is still investigated in numerous works [58, 59, 60].

The Robot Operating System (ROS) includes an object recognition and localisation package, commonly called tabletop perception [61]. It relies on a depth sensor input to segment a horizontal plane, the table, and detect clusters of 3-D points lying on top of the table. To reduce the number of degrees of freedom, from six to four, only objects rotationally symmetrical are considered and they are assumed to sit upright on the table plane. The detected clusters are compared to a database of meshes with an Iterative Closest Points approach. This yields recognition and localisation through the same step. If a cluster does not match any known mesh, a bounding box is computed around the cluster to provide localisation information although no recognition.

In a similar way to the 2-D case, 3-D descriptors can be used to match parts of a scene with known models for detection and recognition. As mentioned earlier, 3-D features have been considered for some time already [30]. However with the popularisation of depth sensors, the variety of descriptors greatly increased [62, 31, 32]. The localisation part is usually achieved through a registration step which aligns the object model with the scene. These approaches have been greatly facilitated by the emergence of the Point Cloud Library (PCL). This library provides algorithms and tutorials for numerous 3-D features and registration algorithms.

Using both RGB and Depth, the Linemod approach relies on robust contours/normals and fast template matching to provide an object recognition and localisation method for textured and non-textured objects. Due to its popularity and because it spans both the 2-D and 3-D worlds, both the OpenCV and PCL libraries implemented this method. More details about Linemod are provided at the end of this Chapter.

The textured, linemod and tabletop methods, plus a method to handle transparent objects, have been grouped in the Willow Garage Object Recognition Kitchen (ORK) project [63].

Though different, all these approaches rely on a perceptual model built from descriptors, for recognition, plus structure and an object frame, for localisation. Oddly, there are few works specifically dedicated to building these models. A notable effort has been done by the RoboEarth project [1] which provides depth and texture modellers plus a web interface to share models. However, it failed to gain widespread adoption. To the best of our knowledge, there is no satisfying out-of-the-box modelling solution and no libraries to easily implement modelling pipelines.

The rest of this Chapter analyses the modelling pipelines and the available tools to build a perceptual model. As explained in the previous chapter, the focus is on perceptual models using textured descriptors.

2.2 Models for Recognition and Localisation

Modelling methods aim at building models as complete as possible. From a perceptual model point of view, completeness means that it provides the data to answer the identity, pose and structure questions under any constraint. The structure information can be interpolated from a point cloud to provide a mesh which gives complete structure information. Every point of the object shape can be expressed from this mesh. We suppose that the frame of an object is located at the geometric centre; so the global frame can be easily defined from the structure.

For textured descriptors, though the idea of interpolating them is appealing, to the best of our knowledge no work proposed such solution. In order to provide descriptors which can describe an object under any circumstance, the model must include as many descriptors as there are variations.

2.2.1 The multiple views paradigm

The appearance of a texture can vary greatly at the pixel level. As the classical textured descriptors work at pixel level, this means the descriptors values can vary greatly depending on numerous variations like lighting, pose, descriptor's repeatability, etc. A solution to this problem is to add as many different views of the object as there are variations, we call this the multiple views paradigm. Two main approaches are used: the online modelling and the virtual modelling.

2.2.2 Online Modelling

In the online method, the camera is moved around an object with concurrent localisation and modelling algorithms allowing to steadily build a model. A first view is taken to initialize the algorithm. As the camera moves, the localisation algorithm computes a confidence score for the current view. When the confidence drops under a given threshold, the last view providing a good confidence is added to the model. As a localisation algorithm is running, the views are registered with respect to some global frame.

This allows adding new views to the model only when it is required. Moreover, it takes into account the current descriptors repeatability, quality of the sensor and lighting. However, if too many errors are introduced during the modelling process, for example due to noise, the whole process has to be restarted. Plus, if a variation, e.g. lighting, should be accounted for, the whole modelling process must be repeated with the different variations.

This makes the method tedious and hardly applicable when complete models are needed. The virtual method alleviate part of these constraints but at the cost of a less complete model.

2.2.3 Virtual Modelling

This approach assumes a textured mesh of the object is available and virtual views can be acquired from this mesh. A virtual camera is moved around the textured mesh and variations are applied. Descriptors are extracted from each view and added to the model. As the virtual camera pose is known, there is no need for a concurrent localisation method.

This solution allows generating models automatically, taking into account as many variations as needed. Moreover, if new variation sources must be added, a new model can be generated effortlessly. However, the textured mesh is captured at one point in time, so sensor's variation or features repeatability cannot be accounted for.

For practical reasons, this is the preferred approach when modelling objects. Though, it implies obtaining a textured mesh of the object in the first place. In the following, we present various approaches to get such mesh and compare their results.

2.3 Obtaining a Textured Mesh

2.3.1 Databases

A straightforward way to obtain textured meshes is to download them from the internet. The communities relying on virtual representations, e.g. for video games or architectural representations, have built huge datasets with textured meshes. To cite a few: archive3d [64], Grabcad [65], Google 3-D Warehoouse [66]. However, most of these meshes focus on illustrating objects and not on exact representations. Though approximate representations can be sufficient for generic level approaches, for instance level recognition and localisation both the texture and shape must precisely match the real object appearance. For this reason, most textured meshes from these sources are not suited for our purpose.

In the computer vision research communities, some works have built datasets specifically designed to assess and compare recognition and localisation algorithms. The Willow Garage dataset [67] gives point clouds and images for 35 objects. A calibration grid is present in the images for pose estimation. In [68], the authors create the RGB-D Object dataset. They take views of objects on a turntable with RGB-D sensors at three tilt positions. The dataset contains 300 objects from 51 classes. This work provides partial RGB-D point clouds with rough localisation data. Though, no meshes are provided. Similarly, Singh et al. [69] use five depth sensors placed at various tilt position above a turntable to acquire views of 100 objects. This work goes as far as to build the corresponding textured meshes. Links to more RGB-D dataset are provided in [70].

As is quickly visible, an important obstacle when using such datasets is that the objects are specific to some countries. Moreover, most of them are commercial objects which packaging changes with time. So these datasets are bound to become obsolete in the short term. For these reasons, static datasets are not reliable and methods to dynamically create textured meshes are needed.

2.3.2 Matching, Motion, Structure

In order to build the textured mesh, the community relies on techniques originating both from photogrammetry [71] and Structure from Motion [72]. These methods allow building the structure of the observed scene, from a large set of overlapping images. By matching point features on two images and using multiple view geometry [57] one can compute the motion between two camera poses. The motion information allows triangulating the matched features to get a 3-D point cloud, further processing can yield a 3-D mesh. In the following these three steps are described in more detail: matching, motion computation, structure computation.

When using local point features, matching refers to the precise coupling of points, from different images, whose neighbourhood describes the same physical area of a scene. Matching is

done by comparing descriptors from two points on an image. In the descriptor space, the distance between two matching points can vary greatly, so it is not possible to use fixed thresholds to determine if two points are matching. Usually matches are determined through a nearest neighbour like approach [73].

Considering two views of a scene, once a list of matching points is determined, it is possible to compute the motion between the views. How to proceed depends on the data at hand. When no structure information is available, one must rely on methods based on the essential [74] or fundamental matrices [75]. When structure is available, one solves the Perspective-N-Point (PnP) problem [58] to compute the motion. These techniques are described thoroughly in Chapter 5. In the particular case where the structure is available but matching points are not, iterative methods, such as the Iterative Closest Point (ICP) method, allow registering sets of 3-D points [76].

To compute the structure, the first step is to have matching points from a set of views and the motion between them. This allows triangulating the 3-D position of each point [77]. The result is a 3-D point cloud. This point cloud can then be processed to obtain a dense mesh representing the object. When the structure is provided by the sensor, the computed motion is directly used to register the incoming cloud with the already built cloud.

The construction of a mesh depends on the available data, whether it is a video or a set of pictures, whether structure is provided by the sensor or must be computed. The next section presents a state of the art of the existing techniques depending on the data at hand.

2.4 Building a Textured Mesh: Practical Methods

Table 2.1: Different methods from the state of the art to build textured meshes.

Ref.	Type	Matching	Motion	Structure	Refine
[78]	SfM	Yes	PnP	Triangulation	Bundle Adjustment
[79]	SLAM	Yes	PnP	Triangulation	Bundle Adjustment
[80]	DS	No	ICP	Sensor	ICP

The basis to compute motion or structure is having matches. This implies having overlapping views of the same scene. A lot of research has explored how to choose good matching views to get precise motion and structure computation, but this problem falls in the domain of planning and is not considered in this Thesis. To the best of our knowledge there are three main approaches for building a textured mesh from a set of RGB or RGB-D images: Structure from Motion (SfM), Simultaneous Localisation And Mapping (SLAM) and Depth Sensor (DS) based.

The Structure from Motion methods use a fixed set of pictures, or views, to create a mesh. The algorithm finds two initial images, the baseline, according to some criterion. Then, the other images are added one by one to incrementally build the mesh. Finally, a bundle adjustment algorithm is run on the available points. As an example, the software Bundler [3] allows the accurate reconstruction of large scenes, like a city [81], in the form of a point cloud. It can be combined with the work from Ponce and Furukawa [82] which allows turning this point cloud into a full 3-D textured mesh. The main drawback is that the views must be taken carefully to have enough overlap and cover the scene properly. Moreover, the dataset quality cannot be evaluated before the whole reconstruction takes place. Recently, Autocad made available a software, 123DCatch [78], which allows creating a textured CAD model of an object from a limited set of views. The mesh reconstruction itself is made offline, in the cloud. Then, the user

retrieves a textured textured mesh. In order to facilitate access for non experts, the software guides the user in finding good views for the mesh building process. But even with this help, getting a good 3-D mesh requires expertise and understanding of the underlying techniques.

The Simultaneous Localisation And Mapping methods build a mesh incrementally by adding new views as they are taken, usually from a video stream. The overlap between views is computed automatically and a new view is added when the overlap is not sufficient for precise reconstruction. Usually, while the reconstruction is under way, a bundle adjustment algorithm enforces coherence among the data. The work from [79] shows an online object reconstruction program using the 3-D mesh under construction to help further reconstruction. They also regularly run a bundle adjustment for increased precision. Royer et al. [83] proposed a similar method for reconstruction but the bundle adjustment is only run on the most recent parts of the reconstructed mesh. The downside of these methods is that they must be executed in one run and restarted in case of problem. More importantly, they can't model objects with discontinuity in the texture as localisation gets lost when no texture is available.

Finally, the last class of methods rely on new generation sensors [84] providing structure information. In this case, matches are computed in 2-D or 3-D images. If matches are between 2-D points, then the motion is computed with PnP techniques. If matches are in 3-D, then the motion is computed using registration methods [6]. Finally motion is used to express the incoming structure, which is in the camera frame, in the global object frame. In [85], the authors present a real time reconstruction method based on intensive GPU use. It computes the motion using Iterative Closest Point to align the structure acquired in successive frames. This kind of approach led to software like ReconstructMe[80] or Skanect[86]. The global frame is fixed and can later be moved to a more convenient pose. The authors of [87] even further simplify the problem by putting the object in a robot gripper. They retrieve the motion from the robot gripper pose and the structure from the depth camera. Table 2.1 illustrates each type of approach with a work from the state of the art and details about the methods.

The SfM and SLAM methods suffer from initialisation problems. As initially there is no structure to compute the motion with a PnP solver, an initial structure needs to be estimated using the fundamental or essential matrix. This asks for complex computations that are useful only once in the whole modelling process. Moreover, the modelling is done up to a scale factor which needs to be determined by including a known object in the scene while modelling. For the DS based methods, they tend to perform poorly on objects with highly symmetrical shapes. Indeed, the registration algorithms need remarkable 3-D points to use as landmarks to align the 3-D point cloud.

2.5 Experiment: Comparing Meshes

This part compares textured meshes created with 123DCatch (123D), a SfM based method, and ReconstructMe (RM) a depth sensor based method. For 123D, the sensor is an iphone camera with a resolution of 1920x1080 pixels. It takes numerous pictures as the object is rotated on a turntable. A maximum of seventy pictures is enforced by the program. For RM, the sensor is a RGB-D Xtion Pro Live sensor with a resolution of 640x480 pixels. The object is continuously tracked while modelled.

Four objects are chosen: a couscous box, a mayonnaise bottle, a milk bottle and a mug. They all exhibit textured and non textured parts. The mayonnaise bottle has a transparent bottom and the mug has specular golden lines on its top and bottom. Table 2.2 shows the resulting 3-D textured meshes for each object.

Table 2.2: From left to right, a picture of the original object, a mesh obtained with 123DCatch, a mesh obtained with ReconstructMe.


BIO
demi-écrémé
BIO
GERMANY
MÜNCHEN

As expected, both methods struggle to represent transparent parts, as can be seen with the bottom of the mayonnaise bottle. They also fail with specular parts as is visible with the top of the mug, though 123D performs better than RM. The front view of the couscous box shows that the 123D method slightly modifies the original texture. In a general way, 123D's texture is close to the original one while RM's texture is blurry. The top view of the mayonnaise bottle and front view of the milk bottle show that the shape on non textured parts is more precise in RM's meshes. Though, the mug handle and top shapes are more accurate in the 123D case than in the RM case.

To summarise for each method, 123DCatch requires one minute to acquire the images for the reconstruction but about one hour to have the mesh created in the cloud. The benefit of this method is that it is possible to use a high resolution camera so the mesh texture is detailed. Though, the texture being combined from different images, it may slightly differ from the object's original texture. The resulting mesh can be precise or fail depending on the features visible at the acquisition time. Non textured parts tend to result in an imprecise mesh.

The ReconstructMe method requires thirty seconds for reconstruction. The process must be done carefully as to not lose the tracking. The resulting shapes are precise as long as there are no specular or transparent parts. Moreover, no texture is needed for the reconstruction. However, when processing highly symmetrical objects, helper features need to be added on the turntable so the underlying ICP can compute the motions. Due to the sensor's low resolution, the textures are blurry and though they can be used to see changes in color or strong contours, details are lost.

The main constraint when using these methods is that they require expertise. For example, the 123DCatch application has a whole page with various videos to show how to acquire a good mesh and how to post process it [88]. In both cases the resulting mesh includes vast parts of the surroundings, so it must be cleaned and the interest part manually cropped out. This is done by using a 3-D modelling software like MeshLab [89] or Blender [90].

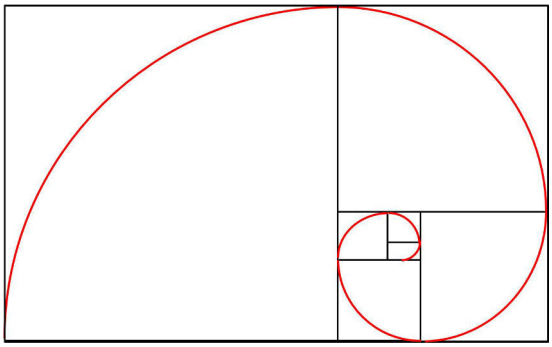
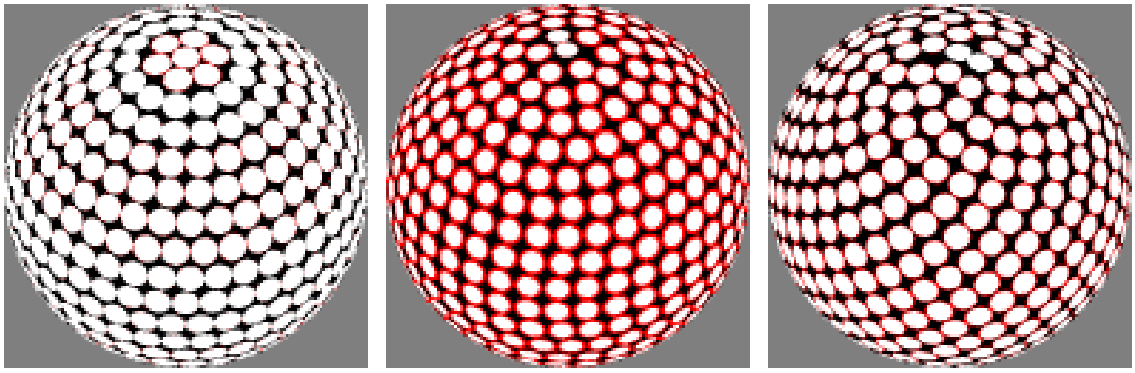


Figure 2.1: A golden spiral is a logarithmic spiral whose growth factor is the golden ratio.



(a) Electrons (b) Simple Spiral (c) Golden Spiral

Figure 2.2: Distribution of 480 points on a sphere with various methods.

2.6 Generating Views

To build a complete perceptual model, numerous views taken from different view points are needed. To ensure homogeneous covering of the possible viewpoints, the views should be as evenly distributed as possible. To do so while retaining the possibility to freely choose the number of views, various methods are available.

Firsts, Rusin proposed a method [91] based on considering that each point of view is an electron and applies a force on the neighbouring electrons. The viewpoint poses are the electrons equilibrium poses (Figure 2.2a).

Saff et al. propose a survey of existing methods to distribute points on a sphere [92]. According to their work, the best approach is to use a simple spiral (Figure 2.2b). The spiral is centred on the object centre and belongs to the plane normal to the vertical axis. One chooses a number of points evenly spaced along the spiral. In order to get tilted views, the spiral plane is tilted with a given step between a minimum and maximum θ angles. Moreover, this method allows sampling the view points in a continuous way while varying the sampling radius. Various independent works found that using a golden spiral (Figure 2.1) avoid having aligned viewpoints when θ varies (Figure 2.2c). The golden spiral method is used in the following of this work.

Table 2.3: This Table shows three parameters (N, θ and R) used to create models for the milk bottle object. The resulting models size, loading time and processing time are compared depending on nine sets of values.

Interval for θ and R (min:step:max)	Sampling points	Number of views	Model size (Mb)	Loading time (s)	Processing time (s)
$-80:40:80$	$N=10$	150	106	18.28	0.18
$0.4:0.2:0.8$	$N=25$	375	261	71.64	0.21
	$N=50$	750	523	143.3	0.25
$0:40:80$	$N=10$	90	63	16.98	0.19
$0.4:0.2:0.8$	$N=25$	225	155	42.54	0.19
	$N=50$	450	309	85.74	0.22
$0:40:80$	$N=10$	60	26	6.6	0.17
$0.6:0.2:0.8$	$N=25$	150	63	16.56	0.19
	$N=50$	300	126	34.14	0.20

2.7 Modelling Pipeline Application

In this Section, the modelling pipeline presented through the Chapter is put to use with a state of the art recognition and localisation algorithm, namely Linemod [35].

Linemod is a multi modal fast template matching method. A set of templates are learned beforehand with various modalities for each template. When an input image is available, local features specific to each modality are extracted. The local neighbourhood around each feature is quantized and the feature is spatially spread by OR'ing shifted versions of the neighbourhood. This step increases robustness to small translations. A response map for each modality is computed by applying a chosen function on the spread features. Finally, a similarity measure between a window on the response map and the known templates is evaluated at various windows positions and sizes. Lookup tables and linearising memory increases the global processing speed.

The main strength of this method is in its capacity to easily integrate different modalities (rgb, depth, sonar, thermal, etc.). Indeed, the modalities responses are combined by simple additions when computing the similarity functions.

In the present case, the modalities are color gradients and normals. As such, shape precision is important while fine textured detail is not essential. For these reasons, we use the ReconstructMe approach to create a 3-D mesh. The 3-D mesh is then sampled thanks to a golden spiral with various parameters depending on the desired model robustness. The parameters are N, θ and R, respectively the spiral number of sampling points, tilt and radius ranges. Finally, each view of the mesh is added as a new template to the final model. As an example, Table 2.3 provides the number of views, model size, loading time and processing time for the milk bottle for nine sets of parameters. The first three sets of parameters cover the object from diverse viewpoints. The second three sets only provide views from above. The third three sets of parameters just afford views from above with a limited radius range.

As expected, when the number of views goes up, the model size, loading and processing times go up. The largest models size is of the order of hundreds of megabytes, which can render this method impractical with hundreds of objects. Especially when considering the loading time: a few minutes per object is unacceptable. For the processing time, it can be increased by up to 28% (18ms to 25ms) for the strongest models. To reduce the number of views, and the complexity,

the modelling parameters need to be strongly restricted, thus reducing the model's robustness. Models light and fast enough to run on multiple objects with a human acceptable speed require a sacrifice to the model's robustness.

2.8 Conclusion

This chapter presented the typical ways to perform recognition and localisation. Because all these methods rely on a model, we focus on the modelling process. One should first choose between online and virtual modelling. Though online modelling captures a realistic model with true variations it is so hard to build in a proper way that virtual modelling is preferable. For virtual modelling, the first step is to build a textured mesh of the object. To do so, we have compared two freely available methods using state of the art techniques. This comparison shows that depending on the localisation method, one should choose the correct mesh building method. For example, for a method strongly depending on object's shape, like Linemod [35], the ReconstructMe solution should be preferred. However, in the case of a method based on local textured feature, 123DCatch provides more precise meshes. Once a textured mesh is available, it is sampled by taking carefully chosen viewpoints in a dense way. However, we believe there are two obstacles when using such techniques.

First, the uniform sampling around an object implies a trade-off between robustness and complexity. Indeed, some parts of the object, the angles of a cuboid for example, require a lot more views as the appearance changes quickly, while the rest of the object can be correctly modelled with a few views. In such case, if the quick changing regions are well sampled, then the rest of the object will be over sampled. On the other hand, if the rest of the object is sampled with just the right number of views, the quick changing regions will be under sampled. With such methods, one must choose between and overly complex model with thousands of views and a model with some parts poorly modelled and possible blind spots. This problem is tackled in Chapter 3 where we try to provide a deeper insight in this compromise.

Second, the modelling pipeline is painful to use. Taking numerous pictures or moving a sensor slowly and carefully around an object is a hard task. Moreover, the resulting mesh is necessarily post processed through a 3-D modelling software. This requires not only time but also advanced expertise in the reconstruction process and in some modelling software. We believe robots should be handled by everyone and as such, the modelling step necessary to make a robot localise objects should also be handled by non experts. The methods described in this Chapter do not allow this. These problems are addressed in Chapters 4 and 5 as we propose simpler methods to build models from images only with, or without, additional information.

Chapitre 3

Dans le précédent Chapitre, nous avons présenté les méthodes de modélisation classiques ainsi que deux de leur principales limitations. Dans ce Chapitre nous traitons du problème de la complexité de modèles crées.

Nous avons vus précédemment que afin de créer des modèles robustes à des variations, l'approche habituelle dans l'état de l'art consiste à acquérir de nombreuses vues incluant les dites variations. Dans ce Chapitre, nous présentons une deuxième option possible : l'utilisation de descripteurs robustes. Plutôt que d'inclure de nombreuses vues, on utilise des descripteurs qui sont inchangé par les variations locales de l'image. Cette méthode permis de réduire grandement le nombre d'images necessaires à la creation d'un model, au détriment de la dimension des descripteurs. En effet, plus un descripteur est robuste, plus ça l'espace dans lequel il s'exprime est de grande dimension.

Les approches basées sur de multiple vues et sur des descripteurs robustes ne sont pas antagonistes mais il est difficile de déterminer l'équilibre adéquat entre nombre de vues et robustes des descripteurs. Dans la suite nous proposons une méthode permettant de se faire une idée de l'impact du choix du descripteur et du nombre de vues sur la robustesse au changement de point de vue d'un modele perceptuel d'objet pour la reconnaissance.

A cette fin, nous proposons une expérience qui utilise la base d'objet RGB-D de Washington. Comme nous nous interessons aux descripteurs texturés nous utilisons les huit objets présentant suffisament de texture pour nos experiences. Pour chaque objet, un ensemble de vues prise à differentes positions autour de l'objet sont disponibles. Sur certaines de ces vues, des descripteurs sont extraits et un modèle est crée à partir de ces descripteurs. Les autres vues sont utilisées pour voir si l'objet est reconnu à partir du modèle crée. Le modèle est crée avec des descripteurs différents dans la premiere experience et avec plus ou moins de vues dans la deuxième.

La première expérience permet de choisir le descripteur le plus robuste aux changements de point de vue. On crée un modèle à partir de deux vues choisies arbitrairement, ce modèle est ensuite utilisé pour essayer de reconnaître les autres vues de l'objets. Un descripteur parfait devrait reconnaître les vues sur des position peu éloignées des deux position présentes dans le modèle. Le nombre de correspondances pour chaque vue forme une courbe avec nombre de correspondances/position de la vue autour de l'objet. Une courbe parfaite est déterminée et la distance de Frechet entre cette courbe parfaite et les courbes générées par chaque descripteur nous permet de détérminer que ASIFT est le descripteur le plus robuste.

A partir de là, en utilisant le descripteur ASIFT, on créée des modèles avec un nombre de vues croissant uniformément réparties autour de l'objet. Chaque modèle ainsi obtenu va être mis en correspondance avec toute les vues de l'objet disponibles. Si le nombre de correspondances dépasse un certain seuil fixé expérimentalement, alors l'objet est considéré comme reconnu depuis ce point de vu là. Le nombre de points de vus depuis lequel l'objet n'est pas reconnu nous informe sur la quantité et taille d'angles morts du modèle. Par ailleurs, la vitesse de reconnaissance ainsi que la taille de chaque modèle sont notées. En comparant les différent résultat on constate que en utilisant 17 vues, on obtient un modèle avec des petits angles morts, une vitesse de traitement faible et une taille réduite. Dans certains cas particuliers, on peut descendre jusqu'à 8 vues en gardant des performances acceptables.

Finalement, comme nous travaillons dans un cadre robotique, nous proposons l'utilisation de modèles « conscients » de leur faiblesses. Si un modèle est capable de connaître ses angles morts et qu'il peut les transmettre à d'autres traitements du robot tels que la plannification, alors le robot peut se débrouiller pour toujours rester dans des points de vues à partir desquels il reconnaît l'objet et ainsi contrer le manque d'information dans le modèle. La conclusion de ce Chapitre

est que en utilisant des descripteurs forts, peu de vues sont necessaires à la construction d'un modèle fiable. De plus, le contexte robotique permet de gérer d'eventuels manques d'information dans le modèle. Cela nous conduit à nous interesser à la construction de modèles en utilisant un nombre restreint d'images, comme nous allons le voir dans les deux chapitres suivants, cela simplifie la construction du modèle et répond ainsi au deuxième problème soulevé en conclusion du Chapitre 2.

3

Complex Models and Robust Descriptors

The cake is a lie.

Doug Rattmann

Current modelling methods aim at perfect models through the use of hundreds of views yielding thousands of local descriptors. As will be seen in this chapter, it confers to the models robustness against viewpoint changes. Like earlier, the term model refers here to a perceptual model. Indeed, in object recognition problems the objects of interest can have any pose, i.e. the object-to-image transform can be of any type, see the transform types Table 3.1. To ensure recognition under any transform, current modelling methods aim for viewpoint invariant models. This is generally achieved thanks to the numerous views of the object and produces complex models. The drawback of such approach is that complex models are hard to build, slow to process and their many descriptors increase the chances of confusion. Though solutions exists to reduce the processing time through efficient matching [93, 94], these methods do not solve the loading time, disc access, storage or confusion issues. We believe that, for many applications, full invariance is excessive and some blind spots in the model are acceptable. Moreover, a same level of viewpoint robustness can be achieved with fewer views by relying on robust descriptors.

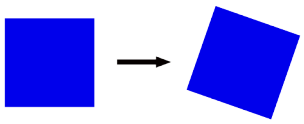
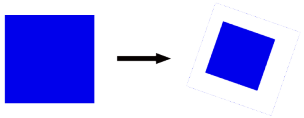
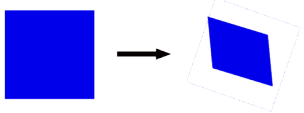
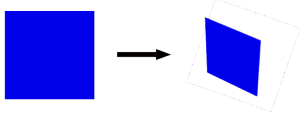
Actually, robustness to viewpoint change can be achieved through two different paradigms.

- Multiple Views: The model can be made robust to a given transform type by including multiple views of the object, under a wide variety of the given transform. For example, to achieve in-plane rotation invariance, the model should include multiple images of the object with different in-plane rotation angles. This method requires numerous views and creates a complex model.

- Robust Descriptors: Some image descriptors already carry their own invariance to given transforms. By using these descriptors, the resulting model acquires the descriptor's invariance. For example, Harris cornerness descriptors [95] are invariant to in-plane rotation. Thus, a model using Harris descriptors is invariant to object rotation in the camera plane. Usually, a descriptor robust to complex transforms has higher dimensionality.

Though the multiple views paradigm is the most popular, this work shows that a correct balance between both paradigms produces light models with high robustness to viewpoint change.

Table 3.1: List of transforms that the image of an object can undergo depending on the viewpoint. The variables t_x, t_y, $r_{i,j}$, s, $a_{i,j}$ and $h_{i,j}$ are respectively translation, rotation, scale, affine and homography coefficients.

Name	Matrix	Figures
Euclidean	$\begin{pmatrix} r_{1,1} & r_{1,2} & t_x \\ r_{2,1} & r_{2,2} & t_y \\ 0 & 0 & 1 \end{pmatrix}$	
Similarity	$\begin{pmatrix} sr_{1,1} & sr_{1,2} & t_x \\ sr_{2,1} & sr_{2,2} & t_y \\ 0 & 0 & 1 \end{pmatrix}$	
Affine	$\begin{pmatrix} a_{1,1} & a_{1,2} & t_x \\ a_{2,1} & a_{2,2} & t_y \\ 0 & 0 & 1 \end{pmatrix}$	
Projective	$\begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & t_{3,3} \end{pmatrix}$	

The contribution of this work is twofold. First, a methodology to assess the robustness of a recognition model is proposed in Section 3.2. Scanning an object observation half-sphere allows finding the blind spots positions and size. Then, this methodology is put to use on 272 models to get an insight into the balance between number of views and viewpoint robustness. Results presented in Section 3.3 suggest that as few as seventeen views provide high robustness while keeping complexity low.

As the multiple view paradigm has been extensively described in the previous chapter, next section focuses on describing the robust descriptors paradigms in more detail and covers the corresponding literature.

3.1 Robust Descriptors Paradigm

Few works have tackled the specific problem of modelling for recognition [96, 79]. In a notable effort, Waibel et al. [1] created a tool to build models by scanning objects. But the resulting models complexity is hard to control and tend to be large. This tool rely on the multiple view paradigm. However, modelling methods can also rely on the robust features paradigm.

Instead of capturing views of the object under different transforms, the robust descriptors paradigm relies on descriptors robust to these transforms. As the descriptors handle the robustness to transforms, a smaller amount of views is required. The following goes over some of the most popular descriptors.

The cornerness descriptor [95] captures the structure of the local neighbourhood with an autocorrelation matrix. It is robust to translations and rotations. The SIFT descriptor [21] relies on histograms of gradients and a multi-scale simulation to achieve high degree of robustness against translation, rotation and scale. The authors of the SURF descriptor [26] use Haar wavelets to approximate the Hessian over a local image patch. The integral image technique allows fast computation. These approaches are robust to translation, rotation and scale. Recently, Yu et al. proposed the ASIFT descriptor [37], an extension of the SIFT descriptor which handles affine transforms. The image being described is simulated under different affine transforms, providing robustness to affine transforms.

These descriptors have high discriminative power at the cost of a high dimensionality. Inspired by the CENSUS transform [97], various works have tackled this problem by focusing on binary descriptors. The ORB descriptor [93] uses differences of pixels to form a binary vector. Proposed by Alahi et al. and inspired by the retina, the FREAK descriptor [98] uses differences of Gaussians to create a binary string.

Finally, the so-called Calonder descriptor [99] is created through a learning process. This descriptor learns, on a set of images, how to be robust against different transforms that are present in the training set. Though having showed good results, it is seldom used. The descriptors cited above and their invariances are summarised in Table 3.2. For a detailed comparison of local descriptors performances, see the work from Mikolajczyk et al. [4].

This section completes the presentation of the two main paradigms available when modelling an object. The multiple views paradigm relies on global data by adding views transformed globally and then computing descriptors on the transformed image. On the other hand, the robust features paradigm relies on local information, by transforming an image patch around the features. According to the work of Lepetit et al. in [100], the former approach is more effective but more complex than the later. In facts, they are complementary, but finding the right balance between the number of views and strength of features while keeping the model complexity low is

Table 3.2: List of some classical descriptors along with their respective invariances.

Name	Rotation	Scale	Affine
Harris	+	-	-
SURF	+	+	-
SIFT	+	+	-
ASIFT	+	+	+
ORB	+	+	-
FREAK	+	+	-
Calonder	Depends	on	training

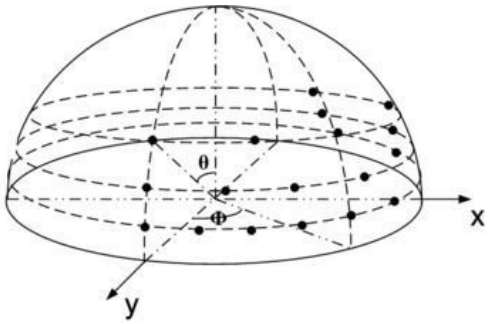


Figure 3.1: The observation half-sphere around an object sampled with angular steps θ and ϕ.

a hard task. In the next section, a robust descriptor is chosen for our experiments. Then various models are compared, with different number of views, to provide an insight about how to find a right balance between robustness to viewpoint change, model complexity and processing time.

3.2 Comparing Models

The general idea is to build a model with a given number of views and a descriptor type. Then scan the observation half-sphere around a test object and find out from how many points of view the object is correctly recognized. Scanning the full observation half-sphere in a continuous way is not possible, so it is sampled discretely as illustrated in Figure 3.1. This sampling method tend to over-represent the sphere's poles. In this work, care is taken to keep some distance between the sampling points and the pole. In the experiments, the minimum θ angle is 30°.

3.2.1 Comparison Method

Consider a set of RGB views of an object, these views form the object set. To create a model, one selects D, a descriptor type, and N views from the object set. These N views form the model set. The model set is for training and the object set is for testing.

Each view in the object set is compared with each view of the model set, using the descriptors D. For each couple of object-model views, matches are extracted and filtered using the fundamental matrix.

For a given object view, the best matching model view is the one with the maximum number of remaining matches. A percentage is obtained by dividing the number of matches by the total number of descriptors in the object view. If the percentage of matches is superior to a threshold, fixed for the whole experiment, then the recognition is considered as successful for this pose. As

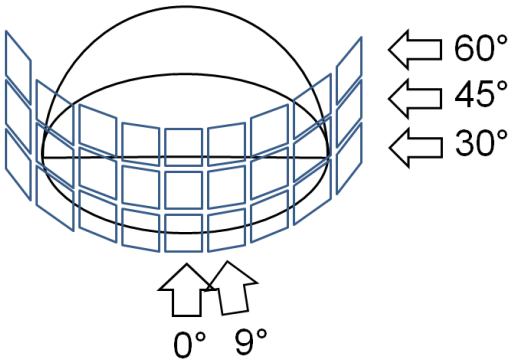


Figure 3.2: The horizontal sampling is approximately 9° with three tilt position, 30°, 45° and 60°.

will be explained later, the dataset provides views with an approximately 9° spacing, so there is no data for some angles. In this case, the number of matches is interpolated by computing a mean between the closest informative points.

3.2.2 Comparison metrics

When building a model for recognition, three properties are desirable: robustness to viewpoint change, low size and low processing time. Viewpoint robustness is measured by quantifying the size of the model's blind regions. The blind regions are the angles from which a camera would not be able to recognize the object, for a given model. The blind region size is measured with the total blind region, i.e. the sum of all blind regions, expressed in degrees. The smaller the total blind region, the higher the viewpoint robustness. The size of a model has no direct impact on the recognition performances but it is crucial when scaling up the number of objects. Models weighting various gigabytes will be hard to store when becoming numerous. It can even come to the point where disc access speed may slow down the overall process. The model size is measured in megabytes. Finally, the complexity of a model impacts the processing speed. It depends on the number of views incorporated in the model, and the dimensionality of the descriptors used to characterise these views. The processing speed is computed in seconds.

The next section describes two experiments to help select a robust descriptor and assess the viewpoint robustness, size and processing time of models depending on the number of views used to create them.

3.3 Balancing complexity and robustness

Some descriptors provide high robustness to viewpoint change but at the cost of longer processing time and/or higher model complexity. Before comparing models, the dataset is presented and the choice of the ASIFT descriptor for this experiment is justified.

3.3.1 Dataset: Washington RGB-D

This dataset comes from the work of Lai et al. [101]. It has been acquired by rotating an object on a turntable in front of a couple of sensors equivalent to a RGB-D camera.

Figure 3.3: An example for each of the considered categories. From left to right: cereal box, food can, instant noodles, soda can, food box, food bag, food jar, water bottle.

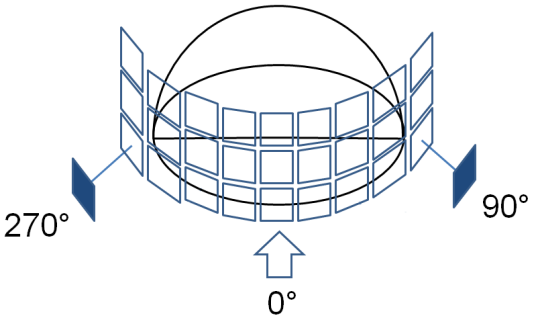


Figure 3.4: The framed squares represent the views from the object set. The plain squares are the views selected to build the model set. In this experiment, the model views are located at 90° and 270°.

For a given object, the camera samples the observation half-sphere with an approximate step of 9° around the vertical axis (pan); and at positions of 30°, 45° and 60° above the horizontal plan (tilt), see Figure 3.2. The resulting dataset contains RGB and depth pictures for 300 objects from 51 categories. Only the RGB pictures are used for this experiment. There are roughly 250 pictures per object, though approximately 160 pictures are labelled with an angular pose. Because texture is necessary for this experiment, only the following categories are considered: cereal box, food bag, food box, food can, food jar, instant noodles, soda can, water bottle. This makes a total of 68 objects from 8 categories, see Figure 3.3. The goal when using various objects is to account for different objects geometries. In this dataset, objects in the same categories have similar geometry. For this reason, the results shown are the mean over the objects from each categories.

3.3.2 Preliminary experiment

The goal of this preliminary experiment is to select the best suited descriptor for the rest of the experiments. To do so, an object is matched with a simplified model made of two views, chosen arbitrarily at 90° and 270°, see Figure 3.4. Five descriptors are compared: ASIFT, SIFT, SURF, ORB and FREAK. Each descriptor is computed on key points obtained with their classically associated key point detector, respectively SIFT, SIFT, SURF, Oriented FAST and FAST [102] detectors.

The model and the object contain two identical views, the 90° and 270° views, two peaks in the percent of matches are expected, corresponding to these identical views being matched, and few matches when moving away from the peaks. Because the objects are not perfectly aligned, at most three successive small peaks are visible, one for each tilt position (30°, 45° and 60°). Note that the curve reaches 100% matches when all points from the three tilt positions match.

In order to quantify the descriptors performance, a model curve is constructed. This curve is

Table 3.3: Frechet distances between the curve generated with the ASIFT, SIFT, SURF, ORB and FREAK descriptors and a model curve. Values are multiplied by 10^{-3} for clarity.

	ASIFT	SIFT	SURF	ORB	FREAK
cereal box	0.73	0.70	0.75	0.72	0.71
food can	0.69	0.95	0.74	0.91	0.85
instant noodles	0.72	0.80	0.73	0.74	0.96
soda can	0.73	0.89	0.78	1.07	0.91
food box	0.69	0.69	0.70	0.63	0.89
food bag	0.74	0.72	0.74	0.78	0.81
food jar	0.71	0.91	0.77	1.09	0.84
water bottle	0.74	0.83	0.75	0.81	0.87
mean	**0.72**	**0.81**	**0.74**	**0.84**	**0.85**

equal to one around $90° \pm 20°$ and $270° \pm 20°$, and zero otherwise. Ideally, the descriptor's curves should look like the model. The Frechet distance from each curve to the model is computed to obtain an error measure. The Frechet distance between two curves A and B is defined as,

$$F(A, B) = \inf_{\alpha,\beta} \max_{t \in [0,1]} \{d(A(\alpha(t)), B(\beta(t))\}$$

A small distance means that the descriptor produces a curve close to the model. The distances for each descriptor and category are summarised Table 3.3. According to the overall mean, ORB and FREAK exhibit similar performances with high error. The SIFT descriptor has medium error. On the other hand, ASIFT and SURF have the lowest errors, though ASIFT error is lower than SURF's.

In order to gain further understanding, the curves for the food box category are shown Figure 3.5. As can be seen, ORB, and FREAK descriptors show numerous peaks and find many matches independent of the angle, thus false matches. These descriptors have the particularity of being binary descriptors, fast to match but with lower dimensionality, thus lower discriminative power. The SURF curve tend to exhibit a high match percentage at angles with no training images, thus false matches. For their part, ASIFT and SIFT descriptors behave similarly showing peaks around the training images positions and low percent of matches elsewhere, but ASIFT tend to fit better the model. ASIFT uses an affine approximation of the object to account for affine transforms. As noted by Lowe in [21], for a given affine transform, the approximation error can be bounded by the projected diameter of the object's circumscribing sphere times a constant. It seems reasonable to believe that SIFT and ASIFT giving similar results hints to the fact that this affine approximation is not valid for objects of this size. Nevertheless, because of its lower error score, ASIFT is chosen to perform the next experiment.

3.3.3 Models comparison

Given the descriptor type and an object, successive models are created using more and more views. The models are compared with the comparison metrics described in the previous section in order to look for the best compromise between model robustness, model size and processing speed. There are 40 views per tilt position, so four models are built using 4, 8, 17 and 33 views uniformly distributed, with tilt 45°. As noted earlier, the ASIFT descriptor is used in this experiment. Each model is then matched to the full object. Two views are considered as matching if at least 30% of their key points are matching, this threshold is chosen empirically.

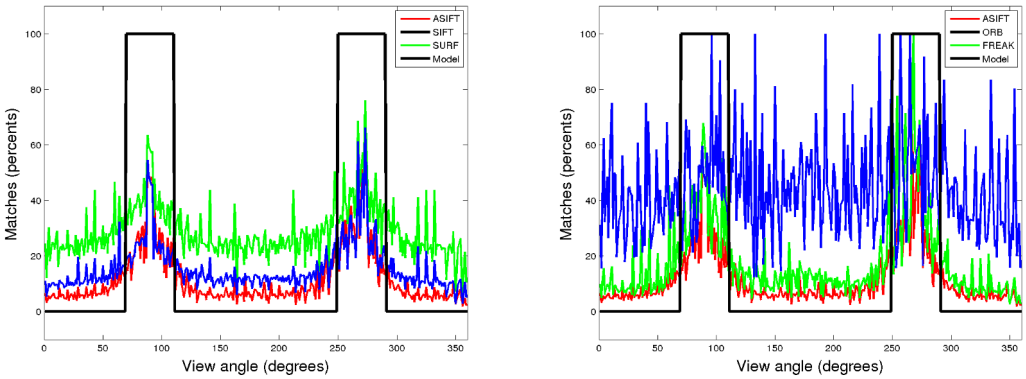


Figure 3.5: Results for the food box category two views model. ASIFT is provided in both curves for comparison.

Key points are matched with a nearest neighbour approach and Lowe's distance ratio [21] is used to discard unlikely matches. Moreover, a RANSAC scheme with a Perspective-N-Point algorithm allows removing matches which are not consistent with a rigid motion.

Table 3.4 presents the results. In terms of blind spots, one should account for the number of views in the model and the fact that there are three tilt positions. Indeed, a sum of blind spots of 100° is likely to mean a $100/3 = 33°$ sum per tilt position. Moreover, the blind spots are in all probability distributed between the model views. So if the model has 30 views, it is likely that there is a $33°/30° = 1°$ blind spot between two views. This seems low enough to be acceptable, especially for applications where moving the camera is possible. Overall, the gain between going from 4 or 8 views to 17 views is higher than going from 17 to 33 views. Plus, using 17 views seems to produce models quite robust as in the worst case the blind spot sum is 224°, which means about 4° between each view. If robustness is to be favoured, using 33 views in the model provides a high degree or robustness, with a minimum sum of 19°, which is 0.19° between two views, to a maximum of 171°, 1.7° between two views. For the particular case of the food can category, a model with 8 views is sufficient for recognition with few blind spots, 3.3° between two views. This may be due to the cylindrical shape which provide features appearance changing smoothly.

With regard to the matching time, it should be as low as possible to allow recognition of various objects simultaneously and at high frame rate. Note that the results shown in Table 3.4 have been obtained with a brute-force nearest neighbour approach. In this particular case, an insight about the asymptotic behaviour can be gained by calculation alone. Indeed, the complexity of the matching algorithms depends on the number of descriptors N, which is closely linked to the number of views, and on the dimensionality of the descriptors space D. Depending on the matching algorithm, having a high N can be preferable to having a high D, or the contrary. For a brute-force nearest neighbour approach, the complexity is $O(DN)$, so a balance can be estimated. Regarding the results, the matching speed scales directly with the number of views in the model. Though matching various objects with a model made from four views is tractable it quickly becomes times consuming when using more views in the models. With seventeen views it is possible to have a recognition time inferior to the second. The high variances in time processing comes from the varying number of key points extracted for each category.

Concerning the size of the resulting model, it is proportional to the number of views. Once again the high variance can be explained by the number of key points extracted for each object.

If models of one megabytes with 4 views are acceptable, it becomes ten times heavier when using 33 views and thus impractical.

3.4 Blind Spots Aware Model for Robotics

When a robot is performing a task which requires to keep in sight some objects, blind spots can hamper the task. In this work, we assume that there are no perfect models. Every model has some blind spots. However, a model aware of its blind spots positions and size allows the robot to plan accordingly. Indeed, if each model provides visibility cones for the object, the robot can plan its actions and motions to keep the corresponding objects in sight. As described in this chapter, the blind spots of a model can be estimated with a virtual representation of the object. We believe models should include their limitations to provide to the robot a better understanding of the world. In this case it requires blind spots aware models.

3.5 Conclusion

This work shows that seventeen views are sufficient to create models robust to out of plane rotation. Added to the translation invariance, in plane rotation robustness and scale invariance of the ASIFT descriptor, it yields models highly robust to viewpoint change. Moreover, for a robotic application where the sensors and objects can be moved by robots and humans, we believe a limited robustness is sufficient if the limits are known by the robot. The small image set compensate for the complexity of the descriptors and overall the models are light and fast to process when compared with models using hundreds of views. Contrary to the current trend, we advocate in favour of light models. More importantly, next chapter will show that lighter models can also be a lot easier to create thus answering to the modelling difficulties mentioned in chapter 2.

Table 3.4: For each considered category, this Table shows the number of views in the model, total sum of blind spots, mean matching time and mean size of the model. The rounded mean and standard deviation over all categories are provided at the bottom row. This experiment considers 360° pan positions and 3 tilt positions.

Category	Views	Blind spots (degs)	Time (s)	Size (Mb)
cereal box	4	243	0.30	2.96
	8	165	0.64	6.59
	17	123	1.36	14.03
	33	99	2.70	26.67
food can	4	216	0.07	0.24
	8	80	0.16	0.48
	17	27	0.33	1.03
	33	19	0.65	2.00
instant noodles	4	323	0.10	0.55
	8	274	0.21	1.39
	17	217	0.45	2.98
	33	170	0.88	5.83
soda can	4	283	0.08	0.28
	8	220	0.16	0.56
	17	188	0.34	1.18
	33	171	0.67	2.28
food box	4	253	0.13	1.27
	8	160	0.28	2.99
	17	100	0.57	6.32
	33	75	1.12	12.26
food bag	4	327	0.21	2.04
	8	295	0.47	4.79
	17	224	0.88	10.31
	33	163	0.58	20.12
food jar	4	205	0.06	0.25
	8	102	0.14	0.51
	17	41	0.30	1.09
	33	33	0.60	2.13
water bottle	4	326	0.10	0.55
	8	286	0.21	1.13
	17	219	0.45	2.41
	33	155	0.87	4.68
mean / std-dev	**4**	**272/47**	**0.13/0.08**	**1.01/0.94**
	8	**198/78**	**0.28/0.16**	**2.30/2.14**
	17	**142/76**	**0.58/0.34**	**4.91/4.58**
	33	**111/59**	**1.0/0.66**	**9.49/8.76**

Chapitre 4

La modélisation d'objets peut se faire avec peu d'images d'après le Chapitre précédent. Partant de ce constat, nous proposons une méthode qui permet de modéliser un objet depuis un petit ensemble d'images. Les seuls hypothèse nécessaires sont les dimensions de l'objet, sa forme (pavé droit, cylindre, sphere, etc.) et les positions approximatives des camera qui ont pris les images.

Le principe est d'utiliser la formule de reprojection de points afin de lever les ambiguités existantes en utilisant la forme et les dimensions de l'objet. Avec cette formule il est alors possible de connaître la position 3-D dans un repère locale à partir de points 2-D visibles sur une images. Avec cette information, on peut utiliser des méthodes classiques de localisation à partir de solvers PnP. La position relative au centre de l'objet de chaque image est donnée approximativement et permet de lier les points 3-D extrait des differents images dans un même repère global propre à l'objet.

Pour mesurer la precision de la localisation avec les modèles obtenus grâce à la technique precedente, on met en place une méthodologie particulère. Elle consiste à placer l'objet dans la pince d'un robot et à mesurer la position du repère de l'objet par rapport à la pince du robot, la transformation depuis la pince jusqu'aux cameras du robot est connu. Il est donc possible de récupérer la transformation allant depuis l'objet jusqu'à la camera. Cette transformée est comparée avec celle calculée par la methode de localisation. Une erreur différente est calculée pour la rotation et la translation.

Pour l'experience en elle même, on met l'objet dans la pince du robot et la pince est placée dans 15 différents positions à des distances allant de 10cm à 1m de distance par rapport à la camera et couvrant tous le champ de vision. Les résultats obtenus sont les moyennes sur l'ensemble des positions. Nous utilisons 15 objets de forme cylindrique ou paralélépipédique, de tailles différents.

Les résultats sont fournis pour chaque objet, mais aussi pour chaque forme d'objets (plan, pavé droit, cylindrique). Ils montrent que globalement les objets sont assez bien pérçus pour être attrapés par un robot. L'erreur en rotation est supérieure à l'erreur en translation. Les cas problèmatiques sont les objets trop petit pour être bien vus à une distance de plus de quelques centimètres et les objets spéculaires dont les reflets faussent la détection de points d'interet.

Dans ce Chapitre nous avons montré que un nombre limité d'images plus les dimensions et forme d'un objet permettent de construire un modele perceptuel facilement. Ce modèle est une approximation, mais nous montrons également que la précision finale est suffisante pour des actions tels que attraper l'objet ou le placer dans le monde. Dans le prochaine Chapitre, nous montrons qu'il est possible de construire un modèle sans les informations de dimensions et forme, pour peu que la méthode de localisation soit adaptée.

4

Simple Modelling Method: Shape A Priori

As shown in Chapter 2, methods aimed at building perceptual models are still highly complex to set up and use. In our opinion, the main challenge nowadays is simplifying this modelling process. In this chapter, we focus on building models as simply as possible, at the cost of model complexity. Indeed, chapter 3 has shown that simple models can provide sufficient viewpoint robustness for some tasks. Loss in viewpoint recognition angles has been quantified and we believe robots can compensate for the model's imperfections. As earlier, we aim at modelling everyday life textured objects. Most of them can be modelled using one image from each side of the object. This means between four (front, back, left, right) and six (+bottom, +top) images. As a consequence, the following chapter proposes a solution to create simple models from a limited set of non-overlapping views with as few human intervention and a priori information as possible.

This chapter presents a framework for modelling objects in a simple fashion when the object shape can be described mathematically. This is particularly useful for simple shape objects like cuboids, cylinders and spheres, as will be seen later. The proposed framework relies on a local frame approach for modelling from non-overlapping images and a back projection method to convert images into 3-D points thanks to an a priori on the object's shape and dimensions. Finally, an evaluation method for localisation pipelines is proposed to estimate the precision of the models obtained through this framework. With this technique, a few pictures, the object dimensions and a shape a priori are sufficient to build a model.

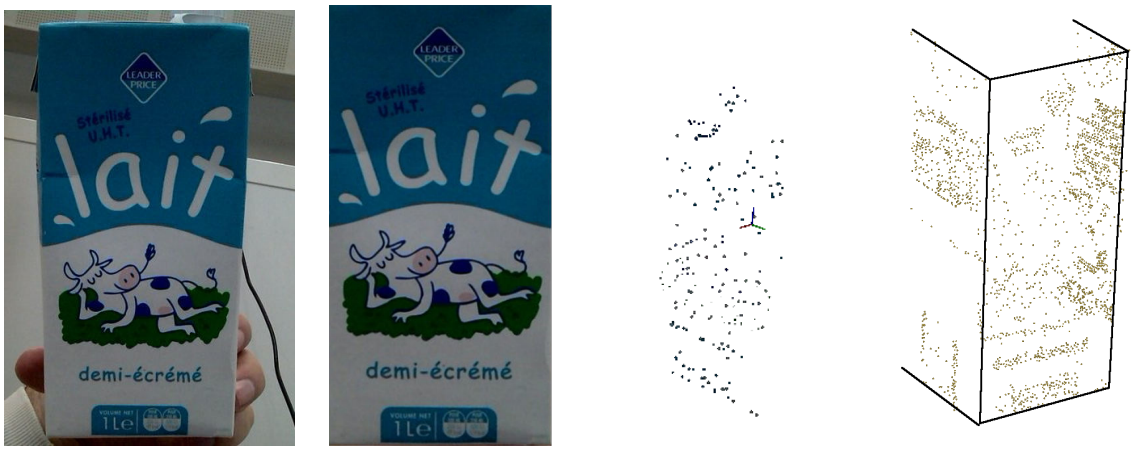
4.1 Parametrisable Objects Modelling

The goal is to build a model for recognition and localisation from the object's shape, dimensions and an arbitrary number of images, with approximative pose, where the object is segmented. For each image available, there are three steps : virtual view generation, image back projection to local frame, local frame to global frame transform.

The first step takes as input an image where the object is segmented. An image of the segmented area is generated as if taken by a virtual camera, hereafter denoted as C_v with known pose with respect to the object. The intrinsic matrix of the C_v is

$$K_v = HK_I$$

where K_v and K_I are respectively the C_v and initial camera intrinsic matrices, H is the projective



(a) Initial (b) Virtual view (c) Local model (d) Global model

Figure 4.1: The three main steps of the modelling framework: (a) initial image, (b) virtual image, (c) local model, (d) global model

transformation from the original segmented region to the C_v image. In order to speed up the process, features are extracted from the C_v image and only the 2-D points corresponding to features are further processed. In the next step, the feature points are back projected thanks to K_v, the object dimensions and a shape a priori.

The back projection equation links the position of an image pixel p to the position of a point in space P. Using the pinhole camera model, the back projection equation can be written as

$$P = \lambda K_v^{-1} R^{-1} p + T, \tag{4.1}$$

where R and T are respectively the rotation and translation from the object frame to the C_v frame and λ is a scale factor. In the general case, R, T and λ are unknown. In the present case, the pose of the virtual camera is known, so R and T are known. In order to simplify further calculus, and without loss of generality, we assume the virtual camera is a fronto-parallel camera, so R is equal to identity and T is a translation along the z axis, $T = (0, 0, T_z)$. In the general case, known constant factors are introduced in the following equations. Using these hypothesis with Equation 4.1, P can be expressed as,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \frac{\lambda}{f_x}(u - u_0) \\ \frac{\lambda}{f_y}(v - v_0) \\ \lambda + T_z \end{pmatrix}, \tag{4.2}$$

where (X, Y, Z) are the coordinates of the point P; (u, v) the coordinates of the pixel p; f_x, f_y, u_0, v_0 the intrinsic parameters from K_v. From the last row of Equation 4.2, the variable λ can be defined as,

$$\lambda = Z - T_z \tag{4.3}$$

where T_z is the translation between the object frame and the camera frame. By definition, T_z is expressed as

$$T_z = -f_x \frac{O_w}{I_w} = -f_y \frac{O_h}{I_h}, \tag{4.4}$$

where O_w and O_h are the object width and height in millimetres, I_w and I_h are the object's image width and height in pixels.

Combining Equations 4.3 and 4.4 and injecting the expression of λ into Equation 4.2 yields

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} (\frac{Z}{f_x} - \frac{O_h}{I_h})(u - u_0) \\ (\frac{Z}{f_y} - \frac{O_h}{I_h}(v - v_0) \\ g(X, Y), \end{pmatrix} \tag{4.5}$$

The only unknown left is Z. However, the shape a priori provides an expression of Z as a function of X and Y. Various examples of $g(X, Y)$ are provided in Table 4.1. Computing $g(X, Y)$ with the expressions of X and Y from Equation 4.5 yields the value of Z. Then, one can compute the values of X and Y.

Hereafter, the calculus for the cylinder case are provided as an example. In this case, g depends only on X and the cylinder radius R, so Z can be expressed as

$$Z = \sqrt{(R^2 - X^2)}$$

By using the expression of X from equation 4.5, one can compute the value of Z as follows,

$$Z = \sqrt{R^2 - ((\frac{Z}{f_x} - \frac{O_h}{I_h})(u - u_0))^2}$$

$$Z^2 = R^2 - (u - u_0)^2((\frac{Z}{f_x})^2 - 2\frac{Z}{f_x}\frac{O_h}{I_h} + (\frac{O_h}{I_h}^2))$$

$$((\frac{u - u_0}{f_x})^2 + 1)Z^2 - 2\frac{(u - u_0)^2}{f_x}\frac{O_h}{I_h}Z + (u - u_0)^2(\frac{O_h}{I_h})^2 - R^2 = 0.$$

This equation can be solved to obtain two solutions for Z. This can be understood by the fact that the back projection equation (equation 4.1) is the equation of a ray passing through the camera centre (u_0, v_0) and the pixel (u, v). This ray intersects with the cylinder shape at two points. The desired solution is the one closer to the camera, which is the negative Z solution.

The resulting set of 3-D points is expressed in a local frame associated with the C_v frame. In order to build a coherent model, the 3-D points need to be transformed from this local frame to a global object frame.

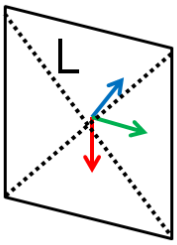
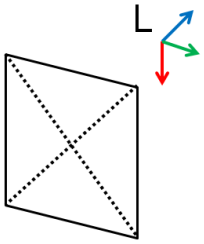
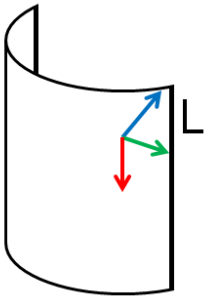
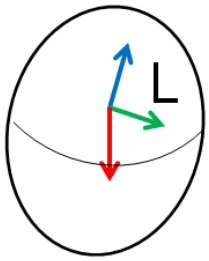
To this end, a global object frame is chosen in a convenient way. Depending on the application it can be the centre of mass, geometrical centre, etc. The 3-D point cloud from the previous step is transformed from its local frame to the global frame, see Figure 4.2. When choosing the global frame it is important to place it at a position where the transform from the local frame is known, for example thanks to the object dimensions. At the end of the process, all 3-D point clouds extracted from the image set are expressed in a unique global frame.

This method is particularly adapted to simple shaped objects which textured parts can be easily described mathematically. Simple objects represent a huge part of everyday life objects like: most rigid objects found in supermarkets, books and magazines, audio/visual supports and packaging, etc. With the object modelling process in mind, the next Section describes how the modelling has been carried out for fifteen objects and how the localisation precision with the resulting models is assessed.

4.2 Model's Precision

In these experiments, fifteen objects have been modelled with the previously described method. The steps required to model an object are:

Table 4.1: For each shape, an example of corresponding local frame L and the surface equation in this frame.

Shape	plane	cuboid	cylinder	sphere
$g(X,Y)$	0	*const*	$\sqrt{(R^2 - X^2)}$	$\sqrt{(R^2 - X^2 - Y^2)}$
Local frame				

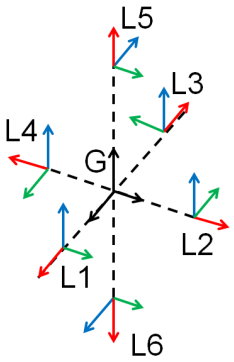


Figure 4.2: An example of local to global poses. The L_i are the local frames and G is the global frame. The frames have been scattered for clarity, in facts they all share the same origin

- Choose a shape approximation for the object.
- Measure the object's dimensions.
- Take pictures of the object from approximately known angles.
- Segment the object on each image.

To keep models simple, each object is described by a single shape: planar, cuboid or cylindrical; there are five objects for each shape. The object's dimensions are measured on the textured parts of the object which will be visible in the modelling images. Then, four images are uniformly taken around the object with an approximate 90° interval between them. This interval is roughly measured with the naked eye. The global frame is chosen as the geometrical centre. Finally, the object is segmented on each image. In this experiment this is done by marking the four corners of the rectangle inscribed in the object of interest. The manual part of the modelling takes a few tens of seconds.

This framework makes the assumptions that the object shape is perfectly represented by some equation, which is not necessarily true. Moreover, the local to global transforms are rough approximations. A set of experiments has been designed to make sure this does not hinder the

Figure 4.3: Objects used in this experiments, from left to right, back to front: mineral water, shaving gel, milk, biscuits, multi-fruit juice, orange juice, olives jar, lentils can, soda can, coffee, newspaper, postcard, teabag, flyer, cards.

localisation process. They also allow quantifying the overall precision of the localisation method in real situations.

Object Pose and Ground Truth

Typically, object localisation methods are evaluated virtually [54], with an offline video [54] or a turn table [22]. However, these approaches limit the number of degrees of freedom of the object pose and as such does not represent all the possible viewpoints. In this experiment, the object is attached to a robot gripper, so it can have any pose in space. Moreover, the poses are chosen to cover as much of the camera field of view as possible, so effects such as radial distortion enter into account. The arm is placed in an arbitrary pose, between 10 cm and 1 m away from the camera. The arm pose may prevent the object from being localised, because of occlusion or because the viewing angle is too high for the view to match the model. As this work focuses on estimating the localisation precision and not the view point robustness, when such case arises a new arm pose is chosen.

The ground truth is retrieved by adding the position of the gripper with respect to the camera, provided by the robot, to the position of the centre of the object with respect to the gripper, measured by hand with a ruler. The full experiment is performed on fifteen objects, see Figure 4.3, with fifteen positions each, for a total of 225 pose estimations.

Error Estimations

To quantify the error, a relative error is computed between the visually estimated pose and the ground truth pose. The error is split into ϵ_r, the rotation error, and ϵ_t, the translation error.

$$\epsilon_r = min(\frac{\|q_{true} - q_{est}\|}{\|q_{true}\|}, \frac{\|q_{true} + q_{est}\|}{\|q_{true}\|}),$$

where q_{true} is the ground truth quaternion and q_{est} is the estimated quaternion. For the translation error,

$$\epsilon_t = \frac{\|t_{true} - t_{est}\|}{\|t_{true}\|},$$

where t_{true} is the ground truth translation and t_{est} is the estimated translation.

For a given object, the results show the mean and variance of the relative error over the fifteen different poses. Overall mean and variance are provided for each shape category and for the whole set.

Technical Details

The previous experiments have been performed on a consumer grade laptop. We use a SIFT detector and descriptor to compute features from the model and current view; these features are then matched by brute force. This is done with the SiftGPU [103] library. Localisation is performed using the EPnP algorithm from OpenCV [55] and refining the localisation with a Levemberg-Marquardt optimisation. The robot is a PR2 and the camera is the left narrow camera of the PR2. The communication between all programs is done by ROS [104].

4.3 Results

The methodology described above is applied to our dataset, and results are summarized in Table 4.2. From a general perspective, the cylinder class is the least accurate. This may be due to the estimate of the local-to-global transform being harder as there are no clear faces. The cuboid and planar classes yield similar rotation and translation error, though planar objects seem to have slightly lower error than cuboids. The non negligible error variance points to the fact that the localisation precision depends on the objects pose, as illustrated in Figure 4.4. It also denotes the fact that for a given object, some parts of the object provide features more robust to viewpoint change than other parts.

In order to get an idea of what the error values represent, Figure 4.4 and 4.5 show the ground truth and estimated frames for various object poses as well as the corresponding errors.

The main error factor comes from objects difference with the a priori shape and specularity. For example, the flyer object has a high error because it tends to deform when held in the robot gripper. For the shaving gel and biscuit objects, imprecisions come from their highly specular surfaces. One can notice that in both cases, it mainly affects the rotation error. In terms of objects size, it does not seem to be a crucial factor. More important is the size of the available textured parts. For example, the cards object performs very well because it as large letters and shapes drawn upon it. On the contrary, the sides of the multi-fruits object performed poorly because the texture is made from small letters.

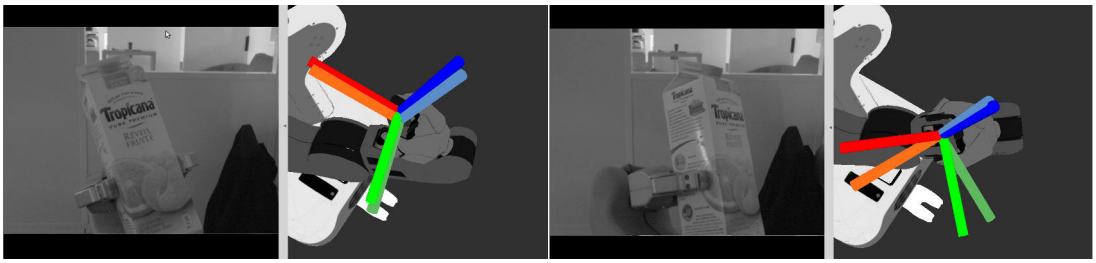
Finally, as can been seen Figure 4.5, the overall mean errors of 5.6% and 4.2% for rotation and translation appear acceptable for many applications like robotic grasping and manipulation.

4.4 Conclusion

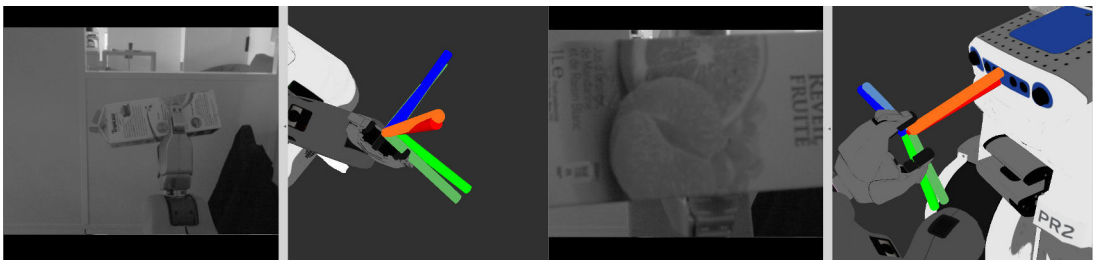
This work shows that an uncomplicated framework, able to model an appreciable part of everyday life objects quickly, provides localisation with a mean rotation error of 5.6% and translation error of 4.2%. This is enough for many applications including grasping and object manipulation. Though, the hypothesis still demand a human intervention to measure the objects dimensions, take the pictures and segment the object. The next chapter introduces a different method which get rids of these human interventions and only ask for simple pictures in the modelling process.

Table 4.2: Mean and variance for rotation and translation errors for the fifteen objects in the test set.

Shape	Object	Mean R_{err} ($\times 10^{-2}$)	Mean t_{err} ($\times 10^{-2}$)	Var R_{err} ($\times 10^{-4}$)	Var t_{err} ($\times 10^{-4}$)
Plane	Cards	4	2	2.2	0.45
	Postcard	2	5	0.7	1.5
	Newspaper	5	4	6.1	3.1
	Teabag	5	5	2.3	1.4
	Flyer	8	3	2.1	2.6
	Mean	**4.8**	**3.8**	**2.68**	**1.81**
Cylinder	Lentils can	8	5	14	6.0
	Soda can	4	5	2.5	2.0
	Shaving gel	7	3	25	1.1
	Olives jar	6	6	15	2.8
	Mineral water	9	5	4.5	7.7
	Mean	**6.8**	**4.8**	**12.2**	**3.92**
Cuboid	Multi-fruit juice	5	5	3.7	3.9
	Orange juice	3	3	1.9	1.6
	Coffee	7	4	17	2.1
	Biscuits	7	5	79	2.6
	Milk	4	3	1.9	0.79
	Mean	**5.2**	**4.0**	**20**	**2.2**
	Overall mean	**5.6**	**4.2**	**11.86**	**2.64**

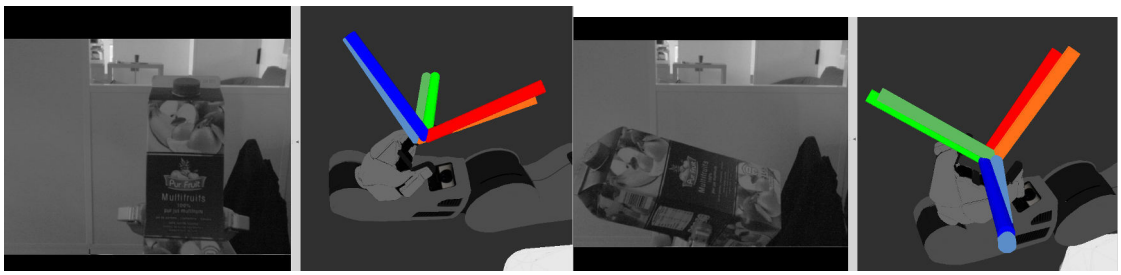


(a) Normal ($\epsilon_r = 0.01/\epsilon_t = 0.03$) (b) Limit angle ($\epsilon_r = 0.11/\epsilon_t = 0.03$)



(c) Far ($\epsilon_r = 0.08/\epsilon_t = 0.02$) (d) Close ($\epsilon_r = 0.06/\epsilon_t = 0.03$)

Figure 4.4: Normal case and three error cases. The ground truth frame is in red-green-blue, the estimated object frame is in lighter red-green-blue. (b) Limit angle, the picture is taken with an out of plane rotation angle at the limit of the features robustness. (c) Far, the object is one meter away, small features are not visible. (d) Close, the object is ten centimeters away, only a part of the object features are visible.



(a) $\epsilon_r = 0.04/\epsilon_t = 0.06$ (b) Limit angle ($\epsilon_r = 0.6/\epsilon_t = 0.04$)

Figure 4.5: Illustration of cases where the rotation (a) or translation (b) errors are higher than the mean errors from this experiment ($\epsilon_r = 0.056$ and $\epsilon_t = 0.042$). The ground truth frame is in red-green-blue, the estimated object frame is in lighter red-green-blue. The object seems to be localised well enough to be grasped.

Chapitre 5

Nous avons vu qu'un modèle peut être construit de manière simple grâce à des images, des dimensions et une information de forme. Dans ce Chapitre, nous montrons qu'il est possible de construire un modèle pour la localisation et reconnaissance à partir d'images seulement si le capteur utilisé pour la localisation est un capteur RGB-D. Les images du modèle, elles, ne sont soumises à aucune hypothèse. La solution que nous proposons consiste à utiliser une nouvelle méthode de localisation qui comprend deux étape de modélisation : hors ligne et en ligne. Cette méthode est testée en simulation et des résultat dans le cas réel sont également fournis. La qualité des résultats nous permet de conclure que cette méthode est une bonne solution permettant de supprimer le problème de modélisation, puisqu'il le modèle n'est constitué que de points d'intérêts extrait d'un groupe d'images ; au-delà de la récupération des images, aucune intervention humaine n'est nécessaire.

Dans une approche de localisation par point d'intérêts classique, il faut disposer d'un modèle 3-D de l'objet à localiser, ou plus exactement des positions des points d'intérêt en 3-D dans un repère lié à l'objet. Ces deux informations, le repère objet et les positions 3-D, sont essentielles et permettent à posteriori en ayant des points 2-D correspondants, de calculer la pose de l'objet d'intérêt.

Dans la présente méthode, nous n'avons initialement pas de repère objet, ni de points 3-D, d'où le terme « structureless ». L'idée est que l'information 3-D provient de la caméra RGB-D et l'information 2-D des images formant le modèle. Lors d'une première étape de modélisation hors ligne, des points d'intérêts 2-D sont extraits. Puis lorsque ces points sont reconnus en nombre suffisant dans une scène, une deuxième étape de modélisation en ligne à lieu. Elle consiste à définir un repère objet selon des règles définies à l'avance (e.g. barycentre des points d'intérêts) et à trouver la transformation entre l'image du modèle et ce repère objet. Dans le cas non calibré, lors du calcul de la transformation modèle/repère objet, les paramètres intrinsèques sont également calculés. Par la suite, chaque nouvelle image est mise en correspondance avec l'image du modèle et la transformation entre cette nouvelle image et celle du modèle est calculée, grâce au points 3-D apportés en ligne par la caméra RGB-D et aux points 2-D du modèle. Cette transformation, modèle/vue courante, combinée à la transformation repère objet/modèle permet d'obtenir la localisation, i.e. la transformation objet/vue courante. Par ailleurs, un repère objet local est crée pour chaque image lors de l'étape de modélisation en ligne et si deux repères sont visibles en même temps, ils sont fusionnés. Cela permet de tendre vers un unique repère objet à mesure que l'objet et vu depuis des points de vu différents.

La précédente méthode est globalement faite en résolvant deux PnP successifs. Afin d'estimer la précision de la localisation résultante, nous procédons à des expériences en simulation. Lors de ces simulation nous varions deux paramètres : le nombre de correspondances disponibles au moment de la localisation et la variance d'un bruit Gaussien appliqué à la position 2-D des points d'intérêts pour prendre en compte les erreur de détection de points d'intérêts. Pour chaque ensemble de points générés, nous comparons notre méthode avec quatre méthodes de l'état de l'art. Nous mesurons la précision des méthodes en calculant l'erreur en rotation et l'erreur en translation.

Les résultats montrent que notre méthode est au moins aussi précise que les solver PnP de l'état de l'art. Par ailleurs on constate que dans le cas non calibré, notre méthode est encore plus précise que l'état de l'art des PnP. Nous attribuons cela au fait que la calibration en ligne revient à déterminer une nouvelle caméra, si du bruit existe cette nouvelle caméra prendra en compte le bruit et absorbera l'erreur qu'il aurait put introduire. Quand le bruit devient trop grand, un position de caméra ne suffit plus pour prendre en compte tous les déplacement de

point d'intérêts dus au bruit et donc l'erreur redevient grande.

Nous concluons ce Chapitre et les méthodes de localisation sur cette approche qui permet de transformer le problème de modélisation d'images en problème d'extraction d'images (depuis Internet, autre) qui, lui, est bien connu. Cette modélisation à partir d'images se fait de manière quasi immédiate et ne demande qu'une rapide étape de modélisation en ligne. Les résultats montrent que la précision est comparable à l'état de l'art ce qui en fait une méthode à privilégier lorsque la modélisation de l'objet doit être faite souvent ou rapidement.

5

Simplest Modelling: Structureless Models

The previous chapter has shown how to build perceptual models for recognition and localisation from a small set of non overlapping images and a few a priori information. In this chapter, we aim at simplifying to the extreme the modelling data by using only unconstrained images and no further information. As explained hereafter, such simplicity in the model calls for particular localisation methods.

Depending on the type of points, 2-D or 3-D, used as model/input, localisation methods can be arranged in four families. In the following, the four families are presented and illustrated through representative works. In the 2-D/2-D case, the model and input points are 2-D projections of unknown 3-D reference points. The classical approaches are based on finding the fundamental matrix [75], in the uncalibrated case, or the essential matrix [74], in the calibrated case. However, these methods lack precision and they may have mathematical flaws which introduce degenerate cases [105]. The 3-D/3-D methods use two different sets of 3-D points, with different frames, as model and input. Such localisation problem can be solved, for example, with the Iterative Closest Point (ICP) approach [76]. In the ICP method, the pose of a set of 3-D points to another is determined iteratively by matching closest points. These approaches are computationally intensive and building a model with 3-D points can prove challenging, even with 3-D sensors.

According to [45], combining 3-D and 2-D data yields greater robustness and precision than those of the two previous families, so the rest of this chapter focuses on mixed families. In the 3-D/2-D case, the model is composed of 3-D reference points and the 2-D input points are projections of the reference points on a camera image plane. This localisation family requires solving the Perspective-N-Point (PnP) problem. Various solutions to this problem will be presented in the next section. The 2-D/3-D case has gained importance with the advent of cheap RGB-D cameras providing 3-D input. In this case, the model is made of 2-D projections of unknown 3-D reference points, and the input is a set of 3-D points in the camera frame. The principal benefit of such approach is that a single image is sufficient to build the model. The main obstacle stems from the fact that no 3-D frame is initially associated to the scene, thus it is not possible to find a motion between the scene frame and camera frame. To the best of our knowledge, there is no localisation algorithm available to directly solve this case.

The localisation methods are based on solving one or various Perspective-N-Point (PnP) problems. For completeness, a brief state of the art of the PnP solvers is proposed in the next section.

5.1 Perspective-N-Point Solvers

When a set of 3-D reference points, expressed in a reference frame, and their 2-D projections on a camera image plane are available, with known calibration, a PnP solver allows retrieving the motion between the reference frame and the camera frames. Solvers can be separated into iterative and non-iterative methods. This work focuses on non-iterative methods as they are faster and thus more appropriate for real-time applications [58].

Nevertheless, this comparison considers one popular iterative method developed by Lu et al. [106] which computes iteratively the rotation and translation using SVD.

In their work [59], Li et al. introduce the RPnP solver. They propose to divide the 3-D reference points into 3-points subsets, express the problem for each subset as a polynomial and then create a cost function from the sum of squares of these polynomials. The solution correspond to the optimum of this cost function.

Zheng et al. suggest a more precise method dubbed OPnP [60]. In this method, the rotation is expressed as a non-unit quaternion, thus relaxing the optimization problem constraints. The whole problem can then be solved with a Grobner basis solver. The main benefit is that it is a global optimization which can handle any singular case and will find all possible solutions. The main drawback being that when various solutions are available, it is not possible to tell which is the correct one. It is interesting to note that in their results, when various solutions are available, the authors select the solution closest to the ground truth in a L2 norm sense. This is not possible for an actual problem.

Finally, the authors of [107] show that a Direct Least Square (DLS) approach can be applied. They propose to use a Cayley-Gibbs-Rodriguez parametrization for the rotation. By relaxing scale constraints, it is possible to express the scale and translation as a function of the rotation. It is then possible to find the rotation with a least square approach and, from it, compute the translation and scale.

The aforementioned methods work well in the 3-D/2-D case. However, they are not applicable directly in the 2-D/3-D case. The next section shows how a 2-D/3-D problem can be formulated as two successive PnP problems and solved with any of the previous solvers.

5.2 Structureless Object Modelling

The 2-D/3-D case has gained importance with the advent of cheap RGB-D cameras providing 3-D input. In this case, the model is made of 2-D projections of unknown 3-D reference points, and the input is a set of 3-D points in the camera frame. The principal benefit of such approach is that a single image is sufficient to build the model. The main obstacle stems from the fact that no 3-D frame is initially associated to the scene, thus it is not possible to find a motion between the scene frame and camera frame. To the best of our knowledge, there is no localisation algorithm available to directly solve this case.

This chapter proposes a localisation algorithm for the 2-D/3-D case, with a calibrated sensor. The problem is expressed as two successive PnP problems and solved with classical PnP solvers. The proposed solution robustness and precision are compared with 3-D/2-D localisation methods based on state of the art PnP solvers.

5.2.1 Localisation with a Structureless Model

In the 2-D/3-D case, no scene frame, nor 3-D points, are initially available in the model, hence the term structureless model. The first step is to create a scene frame S, once and for all, with

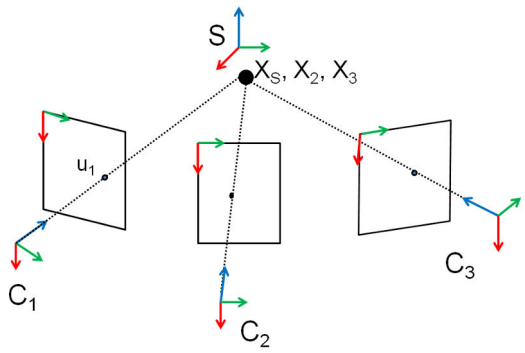


Figure 5.1: Illustration of the proposed localisation method. Though various points are necessary for motion computation, for clarity a single 3-D point X_S and its projection u_1 are considered. The point u_1, on C_1's image plane, forms the model. The input is made of X_2, the point X_S expressed in C_2, and X_3, the point X_S expressed in C_3. First, a frame S is created from X_2. Then, the motion SC_1 is estimated using u_1 and X_2. Finally, the motion C_3C_1 is computed using u_1 and X_3. With SC_1 and C_3C_1, one can retrieve the desired motion SC_3

a first 3-D input. Then, localisation is possible for subsequent inputs.

Consider a set of 2-D points in the image plane of a calibrated perspective camera C_1. Then, a depth camera C_2 provides a set of corresponding 3-D points, for example by matching natural features from C_1 and C_2, with coordinates expressed in C_2. The goal here is to find the SC_2 motion. However, as explained earlier, there is no frame S associated to the scene, so localisation is not possible.

In order to make localisation possible, the 3-D points from C_2 are used to arbitrarily define S. By construction, the SC_2 motion is known. Then, the 3-D points from C_2 are expressed in frame S. Thereupon, the scene has an associated frame and 3-D points expressed in it. With corresponding 2-D points from C_1, it is now possible to use a PnP solver to determine the SC_1 motion once and for all.

When a new set of 3-D points is available from a new depth camera C_3, one finds the corresponding 2-D points in C_1's image plane. These 3-D/2-D correspondences allow solving a PnP problem to obtain the C_3C_1 motion. Since the SC_1 motion is known, computing the SC_3 motion is straightforward. The whole process is illustrated in Figure 5.1.

From a computational point of view, the process can be split in three steps. First, offline modelling, where 2-D points from the first image plane are computed and added to the model. As only 2-D data is required to build the model, it can be built from any camera or from the Internet, provided calibration data is available. Second, online modelling, where a camera input allows defining a scene frame and computing the SC_1 motion. Third, online localisation, where a new camera input allows computing the motion between the scene and this new camera. In the following, each step is described in more detail.

Offline modeling

The starting data is a set u_1 of 2-D points. In the pinhole camera model, u_1 verifies the projective relationship,

$$u_1 = KP_1X_S, \tag{5.1}$$

where K is C_1's intrinsic matrix, P_1 is the SC_1 motion and X_S is the set of 3-D points corresponding to u_1, expressed in the S frame. In this equation, K is known but P_1 and X_S are unknown, they will be determined in the next step.

Online modeling

A new depth camera C_2 provides a set X_2 of 3-D points expressed in C_2, associated to 2-D points u_1 learned in the previous step. A scene frame is arbitrarily defined at the barycentre of X_2 and for simplicity the SC_2 rotation is set to identity. With P_2 being the SC_2 motion,

$$X_S = (P_2)^{-1} X_2, \tag{5.2}$$

This equation allows computing X_S, the 3-D points in the scene frame, from X_2. Then any PnP method allows solving Equation 5.1 to get P_1. The matrix P_1 is saved in the model and represents the SC_1 motion.

In order to get a meaningful scene frame, an alternative is to make a Principal Component Analysis (PCA) of the input point cloud and use two principal axis plus a third orthogonal one as frame.

Online localisation

Now that P_1 is known, it is possible to localise a new depth camera C_3, from which a set of points X_3 is acquired. Let's suppose the correspondences between u_1 and X_3 are known, then adapting Equation 5.2 to this new camera and combining it with Equation 5.1 gives the localisation formula,

$$u_1 = K P_1 (P_3)^{-1} X_3, \tag{5.3}$$

where P_3 is the SC_3 motion, the u_1 are known from the offline model, K is known a priori, P_1 is known from the online modelling step and the X_3 are a set of input points expressed in C_3 frame. Again, any PnP method allows computing $P_1(P_3)^{-1}$. Then, P_3 can be directly obtained.

Note that only K, the intrinsic parameters of C_1, are needed. Calibration data from cameras C_2 and C_3 are not necessary.

Uncalibrated Case

When K is not available, for example when the image is taken from the internet with no EXIF data available, the intrinsic parameters must be determined on the fly. This is done at the online modelling step. Instead of using a PnP solver to recover P_1, an auto-calibration method is used to compute $T_1 = KP_1$. Then, T is decomposed into K and P_1 and the online localisation step can be done normally. In order to compute T_1, the most general approach is the Direct Linear Transform (DLT) [108] which uses a least-square approach to approximate the coefficients of T_1. Finally, there is a direct relationship which allows recovering exactly K and P_1 from T_1. The precision loss due to the approximation of T_1 is evaluated below.

This approach allows using as model a simple image, calibrated or uncalibrated. It enables to use the billions of images available online to build a partial or full model of almost any textured object.

5.2.2 Simulation Results

To assess the robustness and precision of the proposed localisation method, four simulation experiments are set-up. Two of them estimate the robustness against noise and the precision when the number of corresponding points (u_1,X_2) and (u_1,X_3) vary. The third experiment compares the precision of the uncalibrated solution with the precision of the calibrated one.

In order to simulate data as close as possible to real ones, the 3-D reference points should be realistically projected on camera's image planes. To ensure this, the general idea is to define a cuboid in space from which the 3-D points will be picked. Then, cameras are created at random poses such that they see the whole cuboid. Finally, points are randomly picked in the cuboid and projected with noise onto the camera image planes.

The first step is the computation of the minimum distance z, between the camera and the cuboid's centre so that the cuboid is engulfed in the camera's field of view. A maximum distance Z, is chosen arbitrarily while keeping it low enough to avoid the planar case due to distance. To ensure random pose, while keeping the camera oriented to the cuboid, a random point A is picked inside the cuboid and a random point B between the spheres of diameter z and Z centred on the cuboid centre. To obtain the camera frame, an orthonormal basis is created from $\vec{BA}$. This process is done for each required camera, three in this case. Finally, n points are randomly created inside the cuboid and projected on each camera plane with a certain amount of zero-mean Gaussian noise. The points and their projections constitute the simulation data.

In these experiments, the cuboid is a cube of side 2 and $Z = 10$. To be consistent with previous work's experiments [60], in the calibrated case first experiment n is set to 6 while the noise standard deviation σ varies between 0.5 and 5. In the second experiment, $\sigma = 2.0$ and n varies between 4 and 15. For the uncalibrated case, more data is needed to get acceptable precision. In the first experiment n is set to 50 while σ varies between 0.5 and 5. In the second experiment, $\sigma = 2.0$ and n varies between 10 and 100. Each experiment is run 1000 times and the result show the mean errors over all runs.

In the real world case, the test object is chosen with an arbitrary shape. The object is modelled from a single image taken with a calibrated RGB camera. This image is described with SIFT features [21]. The SiftGPU library [109] is used in order to speed up the computation. The matching is done by brute force on gpu and only mutual matches are considered. A Xtion Pro live sensor [110] is used to acquire RGB-D images of the object under different points of view. No calibration data are needed for this sensor. Again, SIFT features are extracted from the images. All points having a NaN depth are filtered out and the remaining ones are matched against the object's model. The first RGB-D view is used for the online modelling step, this step needs to be done only once. All subsequent views can be used for online localisation, they allow computing the object-camera pose. For online modelling and online localisation, transforms are computed using the EPnP algorithm [58], as it is readily available in OpenCV, and refining the result with an iterative Levenberg-Marquardt algorithm. Processing one frame on a consumer grade laptop takes a mean time of 200ms.

For all the experiments, the precision of the proposed methods are compared with the state of the art PnP solvers : RPnP, OPnP, DLS and LHM. For the two calibrated experiments, our approach is based on solving two PnP problems, both are solved using the RPnP solver, as it is the only non-iterative method providing a single solution. Indeed, OPnP and DLS can provide various solutions with no way to discriminate the correct one. For more insight into the working of our method two results are provided: the online modelling error SOM1 and the online localisation error SOM2. For the two uncalibrated experiments, our approach is based on solving an uncalibrated problem, with Tsai's autocalibration method [111] method and a PnP

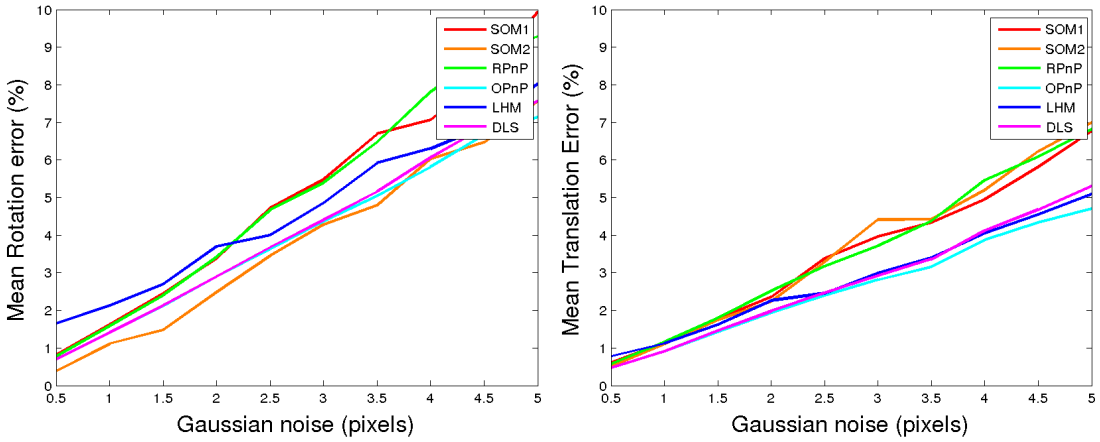


Figure 5.2: Mean rotation and translation errors for six points and a noise with standard deviation varying between 0.5 and 5 pixels.

problem using the RPnP solver, as for the calibrated case.

Calibrated Case

The results in Figures 5.3 and 5.2 show that SOM1, the modelling error, follows the behaviour of RPnP, the underlying PnP solver. Globally, the error decreases with the number of points available and increases with the noise. Regarding SOM2, the online localisation error, its translation error closely follows RPnP performances. However, when increasing the number of points or the noise, the rotation error is lower than the ones of the other solvers.

One could expect our method to perform at best as well as the underlying PnP solver, which is the case for the translation error. However, the rotation error is significantly lower than the one of the underlying PnP method and lower than the ones of state of the art solvers. This could be explained empirically by the fact that our method uses more data than a single PnP solver: a scene-camera transform, a set of 3-D points and a set of 2-D points, are required for the calculus. Moreover, it seems reasonable to think that when a PnP solver is applied to the result of a previous PnP solver, the result is further refined. Finally, the fact that the frame associated to the scene is the 3-D points barycentre may provide some normalisation to the points' coordinates, thus reducing numerical approximations.

Uncalibrated Case

In the first uncalibrated experiment (Figure 5.4), the rotation and translation errors remain low, lower than the PnP solver's for $\sigma < 3.5$. Then, for $\sigma > 3.5$, it quickly grows and at $\sigma = 3.75$ it is higher than the error of any other method. It keeps raising as noise increases. In the second uncalibrated experiment (Figure 5.5), the error is high with less than twenty points. With twenty points and more, the error quickly diminish to an error lower to the state of the art PnP solver's.

Note that none of these methods use a noise reduction approach, like RANSAC for example. However, the autocalibration method computes a new camera intrinsic parameter from noisy points, i.e. which takes into account the noise. The autocalibration process can be understood as finding the image-to-camera pose that minimizes the reprojection error. So the resulting camera suffers from less noise. This can explain why USOM2 has lower error than the other methods. And this points to the fact that autocalibration can be used to diminish noise influence. However,

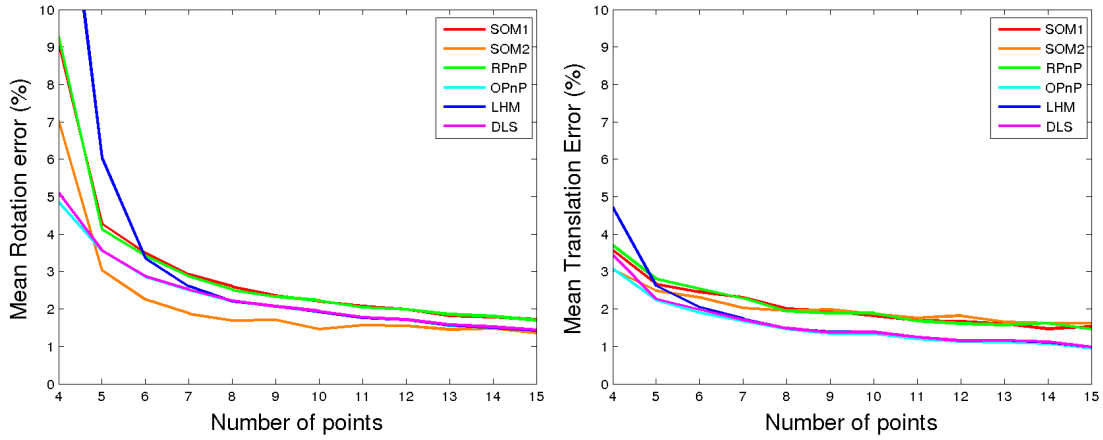


Figure 5.3: Mean rotation and translation errors for a noise with standard deviation of two pixels and a number of points going from 4 to 15.

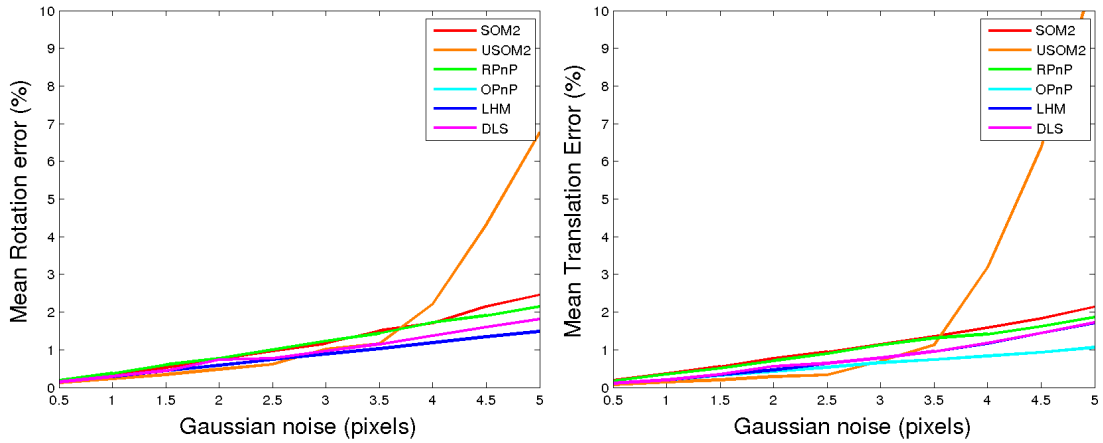


Figure 5.4: Mean rotation and translation errors for fifty points and a noise with standard deviation varying between 0.5 and 5 pixels.

these results also put forward the limits of such methods. Indeed, when there are not enough points ($n < 20$) or too much noise ($\sigma > 3.5$), the autocalibration approach can't find a noise reducing solution.

5.2.3 Modelling with a Calibrated Camera

To illustrate the proposed method in a concrete object localisation case, we proceed to an experiment on regular objects with a calibrated camera. Figure 5.6 (a) (b) are the two images needed to build and complete the model. Figure 5.6 (c) (d) show localisation examples. Note in Figure 5.6 (b) that the object frame which is set up at this online modelling stage is not necessarily aligned with the object shape. Nevertheless, it allows estimating the object motion as long as the current image matches the offline modelling image.

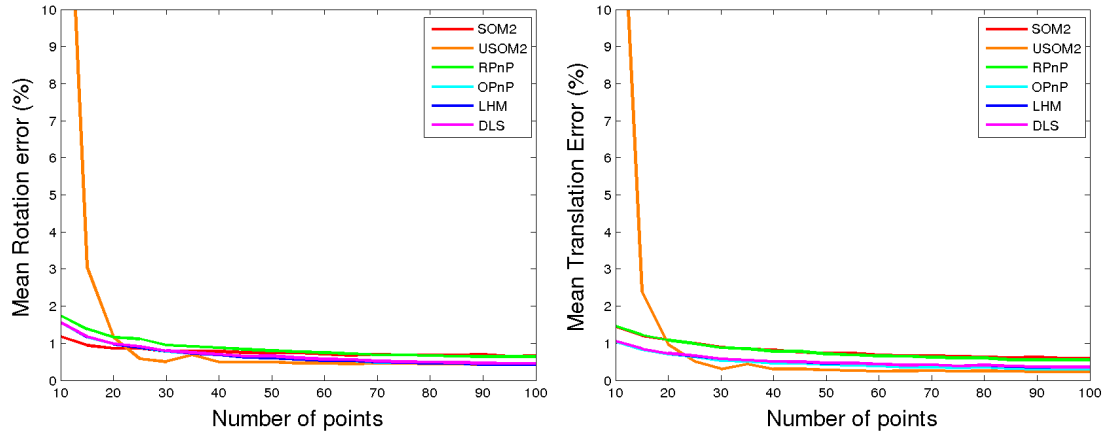
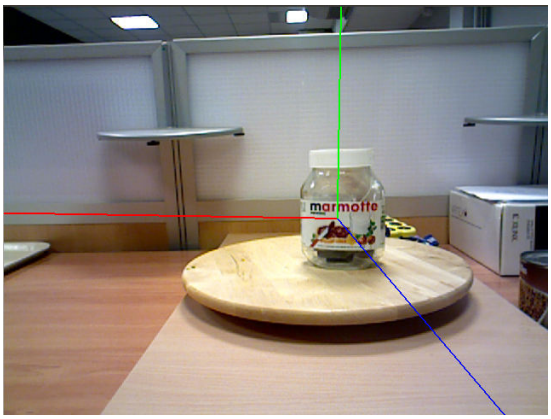


Figure 5.5: Mean rotation and translation errors for a noise with standard deviation of two pixels and a number of points going from 10 to 100.

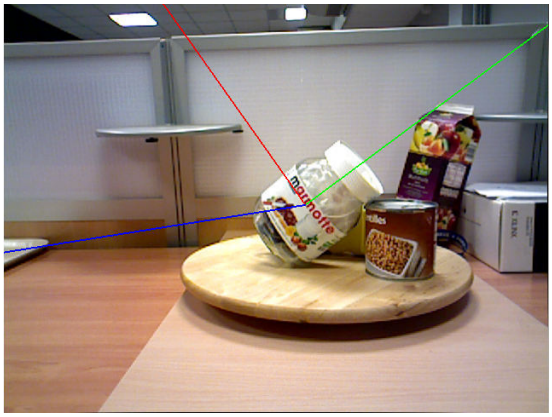
5.3 Conclusion

To the best of our knowledge, this is the first attempt to perform localisation from a 2-D image model and 3-D input. This family of localisation allows using the enormous content available on the Internet to learn objects. Moreover, we believe these methods bridge the gap between the object categorisation and spatial localisation communities. Indeed, object categorisation relies on large bases of images for learning which are usually not usable by modelling for localisation algorithms. With a 2-D/3-D approach, the same data used by the object categorisation method can then be used by the localisation method. The proposed localisation method works with a model made of simple images with no constraints at all. We believe this is the simplest modelling one can hope.

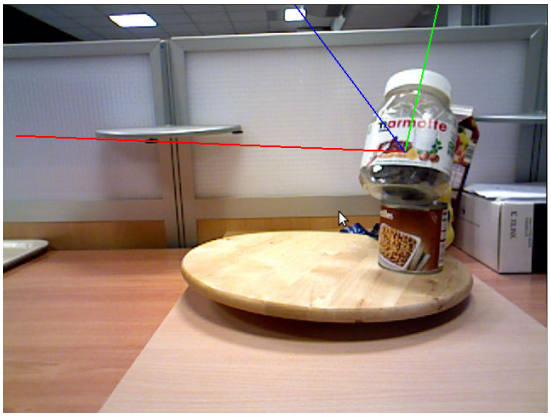
(a) Offline modelling image



(b) Online modelling image



(c) Online localisation



(d) Online localisation

Figure 5.6: Modelling and test images for the marmottela object.

Chapitre 6

Jusqu'à maintenant, nous nous sommes concentrés sur la modélisation pour la localisation et reconnaissance. Nous avons surtout abordés les thèmes de la modélisation et localisation. Dans la suite de ce travaille, nous nous sommes intéressés à l'utilisation du contexte pour faciliter l'étape de reconnaissance.

En effet, dans un cas réel de robotique de service, un robot va devoir reconnaître et interagir avec des centaines, voir des milliers d'objets différents. La complexité de l'étape de reconnaissance, qui consiste à trouver les modèles correspondant aux objets visibles dans la scène, est fortement dépendante du nombre d'objets. Bien que des méthodes visant à simplifier cette étape aient été développées, elles sont souvent spécifiques et la réduction de la complexité vient parfois au prix d'une réduction des performances. Nous pensons que l'utilisation d'information contextuelle peut grandement simplifier l'étape de de reconnaissance.

Dans ce Chapitre nous nous intéressons au contexte apporté par le lieu, i.e. les liens entre les objets, les meubles et pièces dans lesquels ils sont le plus susceptibles d'apparaître. Nous proposons une méthode pour apprendre le contexte lieu par apprentissage. Un robot commence par explorer sont environnement de manière autonome, puis cartographie les lieux et en extrait des informations sémantiques telles que l'organisation des pièces ou la répartition des meubles dans les différents pièces. Puis, à mesure qu'il circule, il apprendre la fréquence d'apparition des objets connus sur les meubles repérés et dans les différentes salles segmentées. Par la suite nous montrons que cette information peut être fusionnée dans une cascade avec des descripteurs visuels faibles. Cette approche permet de filtrer rapidement les objets qui ne correspondent pas à ceux présents dans la scène observée.

Afin d'explorer un lieu, la première étape est de créer une carte 2-D grâce à des méthodes de type SLAM, afin que le robot puisse se localiser dans l'environnement. Cela peut être fait une fois pour toute par un homme télé-opérant le robot. Par la suite, nous présentons une méthode pour créer un modèle 3-D de l'environnement. A partir de la carte 2-D, l'algorithme calcule un certain nombre de points d'observation à partir desquels le robot peut modéliser la plus grande partie possible du site. Le robot se déplace de point en point et observe ses alentours. Il acquiere des nuages de points RGB-D qui sont accumulés dans une carte. Cette première étape permet d'obtenir une représentation du site sous forme d'une voxelmap.

Cette voxelmap est segmenté pour créer une carte topologique des pièces et extraire des zones d'interets, dans le cas présent les surfaces planes des meubles. Pour la carte topologique, on extrait d'abord trois tranches de la voxelmap, elles représentent : le tranche au sol, la tranche qui passe au niveau des trous des portes et fenêtres, la tranche qui passe juste sous le plafond et capture tous les murs. Cette dernière est utilisée pour séparer les différentes pièces. Puis la deuxième tranche, contenant les trous des portes et fenêtres, est utilisées pour localiser les jonctions entre les différentes pièces. Une classification des différents points de jonctions entre deux pièce permet de différencier murs, portes et fenêtres. Ainsi est construit la carte topologique du site.

Afin de segmenter les différentes zones d'interet, i.e. les plans des meubles, la tranche au sol et les points correspondants aux murs sont retirés de la voxel map. Puis pour chaque pièces segmentée précédement, un histogram selon la hauteur est calculé. Les maxima locaux de cet histogram correspondent aux hauteurs ou de nombreux points sont présents, nous faisons l'hypothèse qu'il s'agit là des plans d'interet. A chacune de ces hauteurs, une tranche est extraite et un algorithme de regroupement permet de séparer les différent plans présents à une même hauteur. Le résultat est un ensemble de boites englobantes 2-D délimitant des plans d'interet à différentes hauteurs.

Ces méthodes sont appliquées à l'appartement présent dans la salle ADREAM du LAAS-

CNRS. A l'aide de la carte topologique et connaissant la position de toutes les zones d'interet, le robot circule dans l'appartement et relève la frèquence d'apparition des objets en fonction de l'endroit. Les objets sont déplacés dans une même pièce ou de pièce en pièce, à chaque fois que le robot quitte une pièce. L'ensemble des logiciels sur le robot étant des logiciels expérimentaux, le robot n'a pu identifier que 50

Avec le lien objet-lieu (objet-pièce ou objet-meuble) appris précédemment, cette information peut maintenant être utilisée pour aider à la reconnaissance d'objets. A noter que pour des raisons chronologiques, les résultats de l'expérience précédente ne sont pas utilisés dans les expériences que nous décrivons ci-dessous.

Nous présentons tout d'abord le principe de la cascade pour simplifier la reconnaissance d'objets. Une cascade de descripteurs faibles, et donc rapidement comparables, est mise en place pour éliminer un maximum de candidats dans la liste des objets possibles. Le contexte est également utilisé dans cette cascade et permet de filtrer d'autant plus le nombre de candidats. Les candidats restant peuvent être comparés à la scène par des méthodes de reconnaissance classiques.

Pour démontrer les performances de notre approches, nous utilisons des boites englobantes pour caracteriser les dimensions des objets, des boites englobantes couleur pour décrire leurs couleurs et le lieu comme context. Chacun de ces descripteur forme un étage de notre cascade. Dans cette experience nous considérons 300 objets répartis dans 51 classes. Pour chaque objet, la valeur moyenne et la variance pour chaque descripteur sont calculées, pour le contexte nous déterminons de manière arbitraire des probabilité d'apparaitre dans un certain nombre de lieux.

Chaque objet est observé et on calcule le nombre de candidats supprimés et si le candidat correcte est encore dans la liste des possibles. Les résultat montrent que dans le meilleur des cas, parmis les 300 candidats, il peut ne reste que cinq candidats possibles. Pour les classes, on montre que cela peut aller jusqu'à filtre toutes les classes sauf une sur les cinquante-et-une existantes.

Ces résultats montrent que des descripteurs simples et l'utilisation du contexte peuvent grandement réduire le nombre de candidats lors de la reconnaissance d'un objet, et donc simplifier et accélèrer le processus de reconnaissance.

Nous concluons ce Chapitre en rappelant que nous avons proposé une technique pour apprendre le contexte objets-lieu et pour l'utiliser de manière efficace pour la reconnaissance. L'apprentissage se fait de manière autonome par le robot qui n'a besoin de l'homme que pour la création de la carte SLAM initiale. Le reste du traitement et l'apprentissage des probabilitiés liées au contexte se font de manière entièrement autonome. Pour ce qui est de l'utilisation de cette information, nous montrons que combinées à des descripteurs simples elle peut aller jusqu'à permettre la reconnaissance d'une classe d'objets sans plus de traitements. L'importance du contexte est mise en avant et nous poursuivons dans l'utilisation du context pour la reconnaissance dans le Chapitre suivant.

6

Learning and Using Places as Context

Let us consider a robot designed to help humans at home. It faces an unknown environment, regularly evolving and always unpredictable. In order to operate in such an environment, the robot has to first explore and model it. This model can be a two dimensional map, e.g. for navigation, or a three dimensional map, e.g. for full body motion planning or scene understanding. When a model of the environment is available, given by the user or created by the robot, it must be segmented to extract places and areas of interest. Indeed, typical requests from the user will be "fetch OBJECT from PLACE in/on AREA", where PLACE and AREA are room and furniture names. In order to understand human instructions the robot must be able to associate places with areas and areas with objects. This implies segmenting the environment model in various parts and associating spacial volumes with semantical information. On top of that, humans may lack precision, so a user could just ask "fetch OBJECT from AREA", omitting the place, or even just "fetch OBJECT" with no location specified. In order to handle the lack of information, the environment model segmentation has to provide objects-areas and areas-places links. Though datasets, like the SUN database [112], provide object-place links, being able to learn new links seems crucial for an autonomous robot.

6.1 Learning The Objects-Places Relationship

This Section focuses on robots using directional sensors in indoor scenarios. The next section shows that in this case most existing works propose online exploration and modelling methods. To the best of our knowledge, there are no offline methods. Section 6.2.3 presents such an offline method to find the best 3-D viewpoints to explore a site. Then a segmentation method based on local and global cues is proposed to build a topological map of the site with the different places and how they interconnect. Areas of interest are also extracted and associated to the different places. Finally, section 6.1.3 presents an experiment in a real environment to demonstrate the possibility of learning relationships between objects and areas to help answer user's queries. The algorithms described hereafter are illustrated on the ADREAM apartment (Figure 6.1).

6.1.1 Previous Works

As analysed in [113], an environment can be modelled at various levels: the geometric level, based on features; the topological level, based on views; the semantical level, based on objects and places. Though different, all these problems can be expressed using the SLAM formalism. In [114] the authors show good localisation performances by using a SLAM approach where

Figure 6.1: The ADREAM apartment is used to illustrate the different algorithms and in the experiments described later. Note that the furniture modelled in this views is slightly different from the real setup.

landmarks are known fixed objects and triangulated surfaces dynamically acquired. The authors of [115] recommend the use of a semantical map to enable the robot knowledge to be reviewable and communicable. They use a 3-D laser scanner to acquire a 3-D map through 6-D SLAM. The SLAM considers coarse features, like walls, and finer features, like objects. The work of Aouina et al. [113] shows how to decouple the construction of a localisation model and of a dense 3-D map. The localisation is performed with a 2-D laser range finder while the 3-D modelling is done with a tilting laser.

In the present work, three problems are tackled: site exploration and modelling, i.e. automated construction of a volumetric map; segmentation of the site model into meaningful parts, i.e construction of a semantical map; and the learning of places-objects and area-objects co-occurrence. We provide a quick review of the state of the art for the first two problems: site exploration/modelling and segmentation.

Though this task can appear similar to the modelling of an unknown object by a mobile sensor [116], it is different. In this problem the site may be cluttered by objects preventing the robot from accessing some viewpoints. This problem is also different from the art gallery problem [117]. In the art gallery problem, guards have infinite view range and field of view; this is not the case for a robot.

According to [116] exploration strategies are divided among three types: fixed trajectories, random movements and observation positions. The first approach uses precomputed trajectories to explore any kind of site [118]. Though easy to implement, these methods do not adapt to the site's specificities and can fail for some geometrical configurations. In the random movements approach [119], random points or trajectories are chosen and the robot explores them. This kind of approach has already been rather successfully applied to vacuum cleaner robots [120]. However, it suffers from the trade off between number of random draw, i.e. time spent, and quality of the coverage. Finally, the last type of methods determine the best viewpoints to visit depending on some constraints. Because they are adaptive and robust to the geometry of the site, we focus on these approaches.

In [121], the authors start by acquiring manually a rough estimate of the environment. Then, the model is completed automatically. This method relies on a geometrical approach with vol-

umes representing visibility, mobility and occlusion constraints. In [117], the three dimensional exploration is initialised with the information of a two dimensional map. A voxel grid is filled with *empty*, *occupied* and *unknown* cells. Then, a greedy algorithm is used to select from a set of random viewpoint the one seeing the maximum number of *unknown* voxels. Ray casting is used to test visibility of every *unknown* voxel. A similar approach has been demonstrated at LAAS by Albalate et al. in the framework of the European project CAMERA [122]. A voxel grid is progressively filled with voxels being classified as *unknown*, *empty*, *occupied*, *occluded*, *occplane* (occluded but adjacent to an empty voxel) and *border* (on the border of the line of sight). The next best view is selected based on the number of visible *unknown* voxels and on an estimation of the number of *occluded*, *occplane* and *border* voxels discovered. Finally, in [116], the authors develop a probabilistic framework based on information theory to choose the next best view according to given constraints like travel distance and expected information. In these four works, the aim is to find the best view, acquire it and search for the next best view, and so on.

In the context of this work, a robot has already many critical processes running (motion planning, human perception, actuators control, task planning, etc.). Site exploration is not a critical activity, so instead of taking processing power, we advocate in favour of doing this offline, e.g. when the robot is idle. Thus the process finds all the good viewpoints at once and they can be explored as soon as possible. Such method is proposed in the first part of section 6.1.2.

Once a 3-D map of the environment is available, the robot should segment it into meaningful parts. This segmentation depends on various criteria. In [123], Wurm et al. propose a segmentation algorithm for multi-agent exploration. They segment a 2-D map geometrically with a Voronoi Graph (VG) [124]. The graph is then partitioned by separating clusters at junction nodes which are: local minima, at least of degree 2 (two edges), with at least a neighbour of degree 3 and that lead from unknown to known areas.

The work from Holz et al. [125] elaborates on this method by changing the conditions to choose a critical node. The node must be: close to a Voronoi site, of degree 2, adjacent to a junction node or adjacent to a node adjacent to a junction node (2^{nd} degree adjacency). These modifications provide a better representation of locations such as doorways.

In both cases, the segmentation is based on a geometrical criterion. However, the present work aims at a human representation of the site. In particular, we want to discover the location of rooms and how they are connected by doors and windows. This allows building a graph representation of the site where rooms are vertices and edges represent windows and doors. The segmentation method is shown in the second part of Section 6.1.2. The resulting graph with the rooms and their connectivity is of crucial importance for tasks like object search [126] or learning areas-rooms context, as is shown in the last part of Section 6.1.2.

6.1.2 Exploration, Modelling, Segmentation

To deal with an unknown environment, the first step is to explore it and represent it as a model. In this work, the entities of interest are rooms, areas and objects, so the second step is finding the different rooms and areas from the model. As the objects move around, they can't be extracted from a static model, their occurrences are learned over time. These three steps are described hereafter.

(a) Original (b) 2-D map (c) Raw regions

(d) Reachability map (e) Refined regions (f) Final regions

Figure 6.2: View point selection process. The boundaries of the 2-D map are extended, horizontally in this case. Small and unreachable regions are removed. Regions with similar view points are merged.

Exploration and modelling

The exploration step searches for a set of observation points from which the environment can be modelled. Then, the modelling process aggregates the point clouds found at each observation point and handles occlusions. In the following, it is assumed that the robot has a 2-D map of the environment for localisation purposes and is able to localise itself in the world.

For the exploration step, the world is considered flat, so there is no line of sight occlusion. Starting from the 2-D map, a RANSAC scheme is used to find the obstacle boundary lines and extend them until they intersect with another obstacle (Figure 6.2c). The new boundaries define regions, the centre of these regions form a raw set of observation points. This set is refined by removing regions too small to be of interest. Some parts of the site may not be accessible by the robot. Based on the size of the robot base, a reachability map is created (Figure 6.2d). The points out of reach are also excluded. Finally, regions fully visible from a single observation point are merged (Figure 6.2e). Visibility is tested by ray casting. To find the shortest route along the observation points, the robot solves a salesman problem.

Now, the modelling part assumes the world is 3-D. It starts from the observation points

(a) Voxel map top view (b) Floor slice (c) Ceiling slice (d) Minimum slice

Figure 6.3: The original voxel map (6.3a). The floor slice (6.3b) is removed in further processing to avoid detecting plans at the floor level. In the ceiling slice (6.3c), the walls are clearly visible and the rooms well segmented. The slice where the number of point is minimum (6.3d), shows holes at the door and windows position, plus there is little traces of furniture.

obtained in the previous step. The robot goes from point to point and observes its surroundings.

At each point, a ray casting algorithm classify each voxel as occupied, empty or unknown, if occluded. After this first observation, if large regions of voxels remain unknown, they are represented by the projection of their barycentre on the floor plane. Taking circles centred on the robot sensor with various radii allows searching for a point of view from which the unknown region centre is visible. The robot moves to this point and explores the unknown region. This is repeated until there are no more large unknown regions around the observation point. Then the robot moves on to the next observation point.

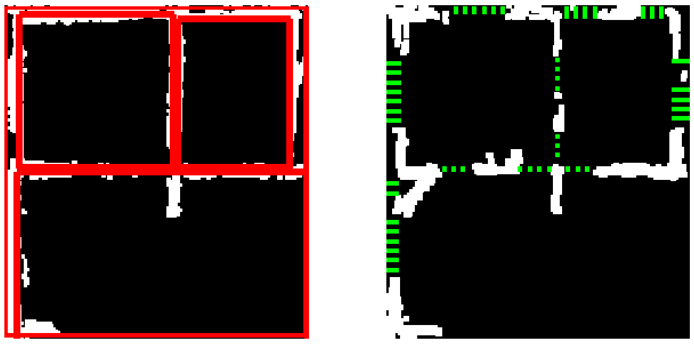
The resulting model is a voxel map containing occupied and empty voxels. The voxel map is built with its Z axis pointing upwards and its X and Y axis in the ground plane. Contrary to hand-made modelling with a depth sensor, this is possible since the robot has a sense of vertical direction.

Rooms and Areas Segmentation

With a voxel map of the environment, the goal is now to extract rooms and areas. A room is defined as an empty space enclosed by walls, while an area is defined as a horizontal surface, corresponding to the planar parts of furniture.

To segment both rooms and areas, we take advantage of the fact that the model is aligned with the vertical axis. It means that some $Z = cst$ planes, hereafter called slices, bear a particular signification. There are three slices of interest. The lower Z slice corresponds to the floor (Figure 6.3b); the higher Z slice corresponds to the walls (Figure 6.3c); the slice with the minimum number of occupied points corresponds to a slice going through the door and windows holes and where the other obstacles are as little visible as possible (Figure 6.3d). To obtain the minimum slice, the histogram along Z of the voxel map is computed. The minimum slice corresponds to the smallest bin of the histogram. In order to reduce noise, each slice is processed with a morphological opening step. These three slices are central for rooms and areas segmentation.

For rooms segmentation, empty connected regions are extracted on the walls slice with a flood-fill algorithm, this yields the different rooms. The contour of each room is represented as the bounding box of the empty pixels belonging to the room (Figure 6.4a). The next step is to



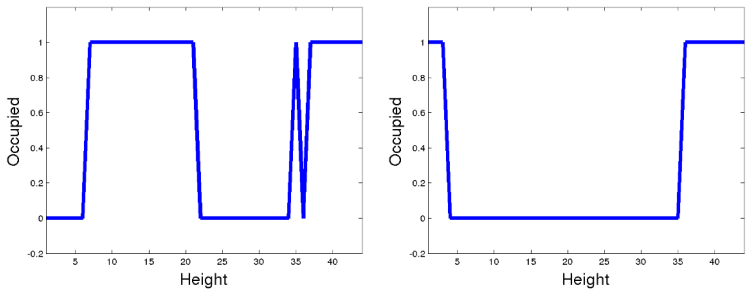
(a) Rooms (b) Bridges

Figure 6.4: The rooms are segmented and their bounding box extracted (red boxes). The exterior of the map forms an additional room. Bridges (green strips) link a room's bounding box to the closest room bounding box, without going through an occupied pixel of the minimum slice. For clarity, only one in five bridges is drawn.

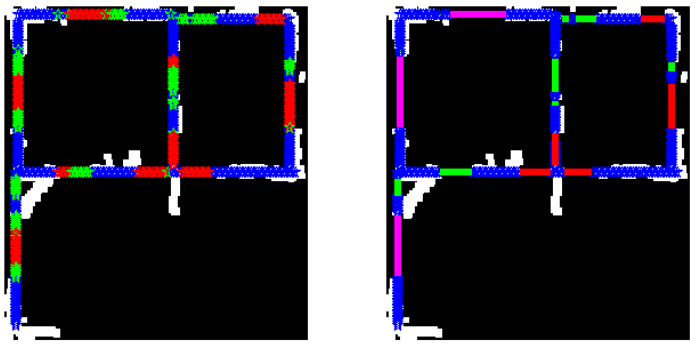
find out how the rooms are linked. This means determining which rooms are linked by how many windows or doors. Bridges are created between the rooms by connecting each point of a room's bounding box to the closest point on any other room bounding box. A bridge is a set of points linking two points from two different rooms. The bridge's points are extracted thanks to the Bresenham algorithm. If a bridge comes across an occupied pixel from the minimum slice, it is to say if a bridge goes across a wall, it is classified as a wall. The remaining bridges connect rooms through doors and windows (Figure 6.4b). Note that the world beyond the borders of the map is considered as an additional room. To classify a bridge as going through a window or a door, we use the Z column of each point of the bridge. The columns are combined with a OR operation to obtain a binary descriptor for the whole bridge. The descriptors are then fed to a hierarchical clustering method to categorise each bridge. The clustering is done by computing the euclidean distance with respect to the clusters medians. The descriptors have specific shapes depending on whether they go across a door (Figure 6.5b) or a window (Figure 6.5a). This results in an initial classification (Figure 6.5c).

With all the bridges classified, the adjacent bridges are grouped into segments. The segments represent whole windows and doors. To categorise a segment, each composing bridge votes for its category (door or window). If the difference in number of door and window bridges is lower than 75%, it is classified as unknown (Figure 6.5d). Small segments are removed as they are likely to originate from noise. Finally, in the same way as for the category, the bridges from a segment vote to determine the rooms connected by the segment. Knowing the number of rooms and the segments that join them, a topological map of the site is created (Figure 6.6a).

For area segmentation, the first step is to remove the floor slice from the voxel map so the floor does not get extracted as an area. The columns corresponding to occupied pixels in the walls slice are also removed so the walls are no longer present. Then, for each part of the voxel map corresponding to a room, a histogram along Z is computed and the local maxima of the histogram are extracted (Figure 6.7). The slices at these heights are the ones with the most occupied points, they correspond to the slices where planar surfaces are present. Slices closer than a certain threshold, $20cm$ in this work, are merged as it is likely only the upper one is



(a) Window descriptor (b) Door descriptor



(c) Initial classification (d) Final classification

Figure 6.5: Typical descriptors for a bridge going respectively through a window (6.5b) and a door (6.5b). Initial classification (6.5c) with the walls in blue, windows in green and doors in red. In the final classification (6.5d), hard cases are labelled as unknown and coloured in purple.

visible. For each of these slices, the connected regions are extracted to retrieve a set of areas (Figure 6.8).

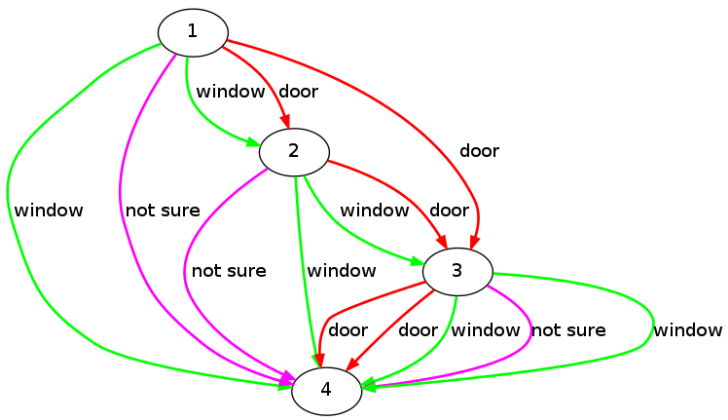
The result of this rooms and areas segmentation stage is a set of places and a topological map with their connections, plus corresponding areas for each room. The areas are represented by bounding boxes with their centre and dimensions.

Learning Object-Areas relationships

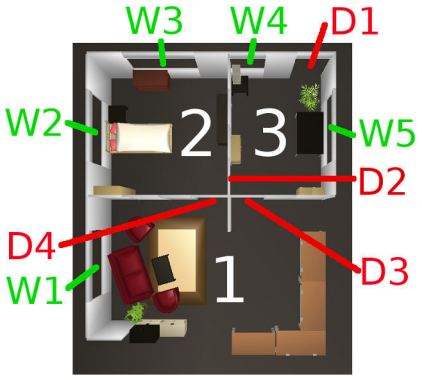
The last stage is to learn objects-areas co-occurrences. As objects locations change over time, the learning must be done as the robot browses through the environment. When the robot goes around, it looks at known interest areas segmented in the previous step. The full area may not be visible in one glance, so the robot should use a scanning strategy. In this work we divide the area using a grid with $50cm$ cells and the robot look at the centre of each cell. Each time a known object is seen, the object-area co-occurrence matrix is updated.

6.1.3 Experiment: Exploring the Environment

For validation, the methods described above are illustrated on a three room apartment staged at the LAAS-CNRS exprimental ADREAM building (Figure 6.3a). The apartment is composed of three rooms furnished with IKEA furniture and as similar to a real apartment as possible. The



(a) Topological map



(b) Rooms numbering and windows/doors labels

Figure 6.6: Rooms are numbered from left to right, top to bottom: 2,3,1. Room 4 correspond to the map's exterior.

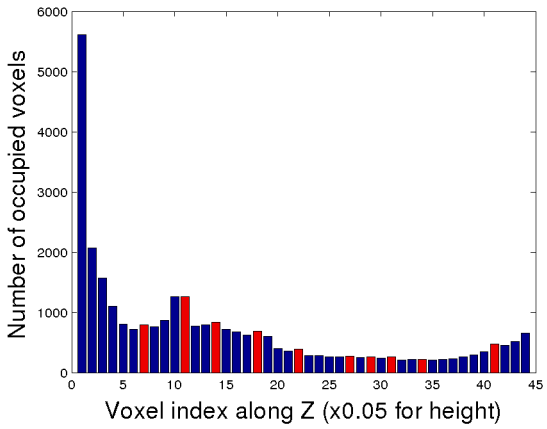


Figure 6.7: Histogram of room 2 with local maxima in red.

robot is a PR2 equipped with a base 2-D laser range finder, a head mounted Kinect and running ROS.

A map of the apartment is built with the laser range finder by teleoperating the robot while running the gmapping package. The localisation is done through the AMCL package which merges the laser SLAM, odometry and map data to estimate the robot pose. The localisation and laser data allow updating a 2-D collision map. Finally, the navigation and collision avoidance is done with the PR2 navigation stack.

For the exploration and modelling, the robot autonomously computes the observation points and models the site. When moving, the robot is localised thanks to AMCL, this allows registering the point clouds from the Kinect and aggregating them in an Octomap [15].

The segmentation step is done offline and a number of rooms and interest areas are found. To estimate the segmentation quality, the number of areas and their height are compared with the ground truth. Area dimensions are not accounted for as they are considered small enough to be handled by the robot scanning strategy. The percent of false positive and false negative are

Figure 6.8: Each red box represents an area. Boxes can be superimposed meaning that there are areas at various height levels.

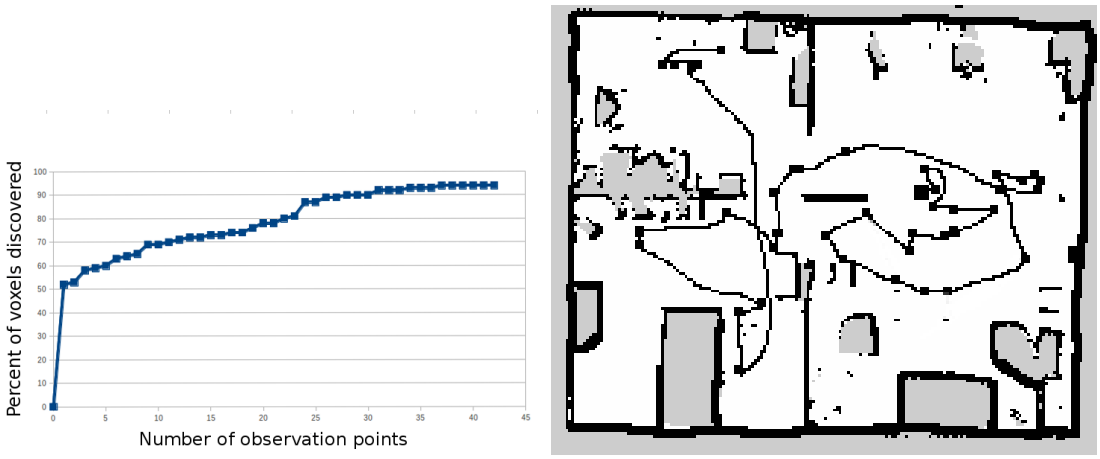
used as error metrics.

Finally, the robot browses through the apartment for three hours and a half and progressively learns which object appears in which area. There are twelve objects with various shapes and sizes: juice, milk, water, wheat, lentils, magazine1-5, coffee and teabag. To keep motion planning simple, the robot moves around in the apartment along predefined points. At each point, the robot stops and searches for areas of interest which centres are at a distance inferior to 1.75m. The area, a bounding box at a specified height, is scanned by small head motions. After each motion a textured object recognition algorithm searches for known objects. If a known object is identified, the co-occurrence matrix is updated. Objects are moved after the robot leaves a room; they are moved inside a room or from room to room. Note that nothing is done to facilitate recognition, so some objects, particularly planar ones, may not be visible when put on high shelves. The resulting co-occurrence matrix and the ground truth are compared and the number of false positive and false negative percentages are computed. The false positive percentage is computed as the number of areas where the object is seen while not being present divided by the total number of areas visited. The false negative percentage is computed by counting the number of times an object is present on an area and is not detected, divided by the number of times the area is visited. These measures do not take into account the case where an object is seen while visiting an adjacent area. So the total error may be higher than the actual error.

6.1.4 Results

The experiment set up in this work is a highly realistic situation where the robot is fully autonomous in a human environment, where objects are moved without consideration for its recognition capabilities. The robot relies on research grade software tools to reach its goals.

Contrary to the preferred full-teleoperation method when modelling a site, in this work the robot is only teleoperated when creating the 2-D localisation map. The environment's 3-D map is created in a fully autonomous fashion.



(a) Discovery curve (b) Trajectory of the PR2 robot during exploration

Figure 6.9: The number of unknown voxels goes down as the number of observation points increase. Note that the trajectory includes the pre-computed observation points and the small motions the robot does at each observation point to explore as many unknown voxels as possible.

The exploration results (Figure 6.9) show that as few as 5% of the voxels remain unknown, these represent the parts of the site where the robot could not go. The map is noisy for two main reasons. First comes the intrinsic precision of the Kinect sensor. In this work, the viewing range is limited to two meters but significant error is already present at this range. Improving on this, for example using a laser sensor, would dramatically reduce the scanning speed of the robot. A second factor is the robot controllers. The head motion when scanning its surroundings is jerky due to the motion controllers. Using soft controllers like in [127] could solve this problem, however to the best of our knowledge, there is no such package available for the PR2 robot. Though the model is noisy, it is currently hard to improve on this aspect without human intervention.

Despite the noise, the topological map created by the segmentation model (Figure 6.6a) has a single error, the window W5 (Figure 6.6b) is mistaken as a door. This is due to the fact that a table in front of the window prevents the model from seeing the wall part under the window. Moreover, when computing the bridges descriptors, the table is barely visible, it corresponds to a peak at a specific height which is not differentiable from noise.

In the three other cases where there is a piece of furniture below a window (windows W1, W2 and W3), preventing points from being acquired there during the modelling step, our method finds the ambiguity and classify the segment as unknown (Figure 6.10).

For the interest areas segmentation, the main goal is to obtain as few false negative as possible, at the cost of false positives. It is preferable to scan useless areas than to miss interest areas with objects on them.

Results shows that there are no false negative (Figure 6.8), i.e. the 38 interest areas in the apartment are found, though there are 46% false positives. Roughly half of them are due to noise on partially reflective surfaces, screens in this case, introducing enough noise to make them appear like a narrow planar area. These could be removed with close inspection when the robot goes through the environment. It is not done here as the goal of this work is to have a purely offline segmentation.

(a) Living room

(b) Bedroom

Figure 6.10: Three situations where the segmentation is not sure about the segment category. Each time a piece of furniture is blocking the view: the couch (6.10a), the bed and the chest of drawers (6.10b).

Finally, segmentation groups objects close together, creating big areas which are not well represented by bounding boxes. The robot scanning strategy partially handles these areas.

Objects	FP	FN
Juice	3%	14%
Milk	4%	14%
Water	2%	30%
Wheat	6%	0%
Lentils	1%	57%
Magazine1	4%	14%
Magazine2	3%	29%
Magazine3	0%	86%
Magazine4	0%	43%
Magazine5	0%	71%
Coffee	1%	86%
Teabag	0%	71%
Mean	**2%**	**43%**

Table 6.1: False positives (FP) and false negatives (FN) for each object.

For the learning results, Table 6.1 sums up the percentage of false positives and false negatives for each of the twelve objects in the apartment. It can be seen that large objects, like Milk or Juice, have a lower false negative rate than small ones like Coffee and Teabag. For the magazines, their planarity makes them hard to detect especially when placed on high shelves, except for the Magazine1 which sits upright. A large part of the errors come from the simple recognition models used in this work. As any model, they have limited scale and viewpoint robustness making some recognitions hard. The false negatives also come from the scanning strategy which struggles for areas where a bounding box is not a good representation, like area in front of the W1 window, composed of the couch and shelves (Figure 6.10a). Eventually, both errors would be reduced as the robot continues going around and learning over time. If some objects are missed, they can be seen later.

6.1.5 Conclusion

This work demonstrated that a robot can autonomously explore, model and segment a site in a realistic situation. There are no hypothesis on the rooms except that a bounding box should be a good approximation of the rooms and areas shapes. More importantly, exploration and segmentation do not require rooms to be aligned or in any particular arrangement. These experiments even go further by exploiting the segmentation to learn Areas-Place and Object-Areas co-occurrences. Again, no hypothesis is made on the objects or areas and how they are arranged, except that the object's models are known. For the objects detection, the high false negatives rates can be explained by the difficulty of the task at hand, the realism of the experiment and the quality of the software used. Future works involve adding online corrections to the process, so places and areas extracted during the offline segmentation are refined online, and relying on a more sophisticated recognition to reduce the false negatives rate. With places-objects and areas-objects relationships learned, they can be used to improve object recognition as explained in the next Section. Note that because of chronological constraints, the work described hereafter does not uses the results obtained earlier.

6.2 Recognition knowing Objects-Places Relationship

As seen in Chapter 3, to achieve robustness, objects descriptors have high dimensionality. However, high dimensionality descriptor matching implies a high computational cost. Moreover, the complexity grows with the number of models in the database. In a real application, the robot needs to recognize tens of objects at a time and potentially thousands of different objects. The following shows that classical approaches fail in these cases.

As seen in the first Section of this Chapter, the robot can access more information than just camera inputs. It is located in space and time, it can be given prior knowledge. The robot can put a context on what it perceives. Incorporating the context to the matching process can be a solution to discard unlikely models so more complex methods can then be applied to a limited set of candidates. Moreover, we advocate in favour of combining simple visual cues with the contextual information to filter out as many false candidates as possible.

In the following, the term (known) model designate the descriptors stored in the database of known objects. The candidates are the remaining models after preprocessing. Finally the term (unknown) object represents the parts of the observed scene segmented as potential objects yet to be identified. Next section provides a reminder of the state of the art of visual descriptors and shows that they tend to have high dimensionality and thus high complexity.

6.2.1 Related Work

Historically, the first object recognition approaches exploit the image edges (internal or silhouette) to create a model [128]. However, these descriptors are sensitive to illumination changes, noise, blur and occlusion. In order to increase their robustness, [129] propose using the Distance Transform (DT).

In [21], the author uses points instead of edges. The points are described with the Scale Invariant Feature Transform (SIFT). This point-based method is robust to illumination, rotation and scale changes. Moreover the point-based methods tend to be resilient to occlusions. However these methods get confused when similar textures are observed [40].

Recently, the introduction of cheap and easily accessible 3-D sensors (Kinect, Xtion, etc.) has led to the design of 3-D descriptors [5]. In [31], the authors propose the Signature of Histograms

of OrienTation (SHOT) 3D descriptor, similar to SIFT but generalized to the 3D case. In [62], the authors introduce the Viewpoint Feature Histogram (VFH) descriptor which is related to the viewpoint. This generation of descriptors are robust to most variations but their computational cost is still too high for a real-time robotic application [5] [68]. Some of the main descriptors from the state of the art are presented in Table 6.2.

Table 6.2: Descriptors from the state of the art and their dimension along with the proposed descriptor.

Ref.	Descriptor	Dimension
[128]	Shape context	60
[21]	SIFT	128
[31]	SHOTS	32
[62]	VFH	263
Ours	Minimum Volume Oriented Bounding box	3

Regardless of their robustness, these approaches suffer from an increased complexity as the number of learned objects grows. A solution proposed in [129] is to organize the descriptors in a tree, for a given class, in order to reduce the search cost. In [130] the authors propose to group the classes which share identical features so the number of candidate classes diminishes with each processed feature. Finally, in [68], the authors create a tree containing the classes, instances and poses for various objects. These methods allow speeding up the matching but their complexity remains strongly linked to the dimensionality of the descriptor used.

From a different perspective, the work of Divvala et al. [9] show that different contextual sources can be used to increase a categorisation task precision. However, in assistance robotics we believe speed is more important than precision for the robot to be accepted by the human. As a consequence, we propose to use the context to increase the categorisation speed, rather than its precision.

As noted in [10], contextual and visual information can be mixed successfully with a cascade of boosted classifiers. We propose to adopt a cascade of classifier mixing visual and contextual data. The visual processing steps are lightened by relying on simple visual descriptors. This preprocessing step quickly discards models different from the observed objects or that do not fit the context. Choosing among the remaining candidates can then be done with a classical approach at a lower computational cost.

6.2.2 A Cascade of Minimum Volume Bounding Boxes

Cascade of weak descriptors

As demonstrated in [131], a cascade of weak descriptors can quickly discard a huge amount of unlikely classes in a classification process. In order to keep high recognition rates, each level of the cascade must have a very high true positive rate even at the cost of a high false positives rate. More levels in the cascade means a higher detection rate is needed at each level to avoid false negatives. Indeed, each level adds a chance to make a mistake. But each level also discards more unlikely classes. In order to operate quickly, the levels of the cascade rely on low dimensionality descriptors or on learned contextual probabilities.

Minimum Volume Bounding Boxes

In this work we choose as low dimensionality descriptor the dimensions of the Minimum Volume Bounding Box (MVBB). The MVBB is a the minimum volume virtual box containing all the points of an object's point cloud or color cloud. In geometrical terms, it is a transform from the cloud frame to the MVBB frame plus a three dimensions vector containing the size of the box. The box frame does not have to be aligned to the cloud's reference frame. The MVBB computation is described in section 6.2.3. The MVBB offers more information than axis aligned bounding boxes as it gives orientation in addition to dimensions and translation. Moreover, it is robust to rotation, scale and occlusion to some extent. For a given object, the spatial and color MVBB are computed. For the color MVBB, the input is transformed to the Lab color space and a MVBB is computed in the color space. The dimensions of the MVBBs are organized in a vector in descending order. The biggest value is the first element of the descriptor vector, the smallest value is the last element of the vector. This implies that there is no distinction such as width or height of an object. Though less discriminative, this gives the descriptor robustness against object pose variation. Finally, in order to be robust to viewpoint change, the MVBB is computed from different points of view for a given object. The final descriptor is thus composed of the mean and variance for each dimension of the MVBB.

Place as Context

As explained earlier, the places and areas observed by the robot can be used as context. For simplicity, and because it does not impact the reasoning, in the following the term place is used for both places and areas. For each model, a probability to find the object is assigned to each place. The probabilities are uniformly distributed to provide a ranking information. In order to account for exceptional situations, e.g. a toothbrush in the kitchen, all places have a non null probability.

Adjusting Recognition Confidence by chi-square test

Provided a set of training descriptors for a known model, the mean and covariance matrices are computed for each dimension of the descriptor. When a new object is seen, its descriptor is computed. Then the Mahalanobis distance between this descriptor and the training set mean/covariance is computed. We make the assumption that the object's dimensions follow a Gaussian distribution. In such case, the Mahalanobis distance follows a chi-square distribution. So it is possible to know if the object matches a known model with a certain level of confidence. Requiring a high level of confidence means having fewer false negatives but more false positives.

6.2.3 Experiments

The MVBB are obtained with the algorithm from [132]. It computes the MVBB with some error epsilon, thus allowing a lower complexity $O(n \log n + n/\epsilon^3)$ than others approaches. The output of this method is a rotation, a translation from the point cloud frame to the bounding box frame and the three dimensions of the box. This algorithm can fail for some configurations, for this reason some examples in the dataset have been discarded. Moreover, examples yielding a MVBB with one of its dimensions null have been discarded as well. For each model, the probability to appear in a given place depends on the object class and on common sense. The places used in this work are: kitchen, dinning room, living room, bathroom, bedroom, garage and outdoors.

For chronological reasons, the results showed in the previous Section are not used and values are assigned manually.

The considered objects come from the Washington RGB-D dataset [68]. This dataset contains more than 100.000 RGBD point clouds with various poses for 300 different instances from 51 classes (Fig.6.11). Please note that the classes are organized semantically by hand and not by some objective criterion. As can be seen later, this explains some difficulties in a classification task. As this dataset contains a lot of information, we set up three experiments described hereafter; the results are available in the next section.

Figure 6.11: One object from each class of the dataset.

The first experiment is designed at countering the intuition that the spatial bounding box of an object strongly depends on the point of view. A MVBB is computed for each pose of each instance. The mean and covariance matrices are computed for each instance. For clarity, only the three instances with highest and lowest standard deviation are presented. Results are shown for spatial and color MVBB in Table 6.3 and 6.4.

The second experiment is aimed at estimating the efficiency of a cascade of weak descriptors as a preprocessing step. It discards numerous unlikely objects from the database, leaving few candidates for a subsequent stronger recognition step. In the second experiment, we use a cascade composed of a spatial MVBB descriptor, a color MVBB descriptor and a contextual step based on the object's place. The experiment is done at class (resp. instance) level. The dataset is split in a training set, 75% of the instances (resp. poses), and a testing set, 25% of the instances (resp. poses). At test time, a place associated with the current object is randomly chosen according to the co-occurrence probabilities of the object. For each candidate pose (resp. instance) the Mahalanobis distance to each class (resp. instance) is computed. Then according to the confidence required, 90%, 95% or 99%, the number of candidates discarded by the cascade is computed.

Finally, the cascade is tested as a standalone object classifier. For practical reasons, the context is not considered in this experiment. The precision-recall curve is computed in a one-vs-all manner to show the performance of this small cascade. This is done as an exploratory experiment with a view to enlarging the cascade for better results. For clarity reasons, six classes representative of the dataset are chosen empirically.

Table 6.3: Three lowest and highest standard deviation, in meters, of spatial bounding box when pose changes.

Instance	std X	std Y	std Z
peach_1	0.00267756	0.00164023	0.00339322
bowl_1	0.00299135	0.0022616	0.00351028
orange_2	0.00263584	0.00221951	0.00382553
shampoo_2	0.0566599	0.020969	0.0135131
keyboard_5	0.0734937	0.0299407	0.00467566
binder_3	0.0622888	0.0966164	0.0140562

Table 6.4: Three lowest and highest standard deviation of Lab color bounding box when pose changes.

Instance	std X	std Y	std X
orange_3	3.0864	4.26667	4.16493
garlic_2	5.49994	3.35492	2.33083
pear_7	4.07947	3.55293	4.39076
shampoo_2	28.446	24.1258	11.2619
binder_3	34.7907	17.6257	14.5281
dry_battery_6	26.7826	25.3871	19.8527

6.2.4 Results

In this first experiment (Tables 6.3 and 6.4), we see that the standard deviation for the spatial MVBB is small, a few millimetres in the best case and some centimetres in the worse case. Even in the worst case, only one dimension has a variance that exceeds 6.3 cm; the two others tend to be small. For color MVBB, it is difficult to assess the difference between colors. For comparison, Figure 6.12 show some colors separated by the standard deviations of Table 6.4. It is notable that some objects have uniformly distributed colors which yield small standard deviations, e.g. natural objects like lemon, apple, etc. However, other objects vary strongly in color from one viewpoint to another, e.g. manufacture products, packaging, etc. The color MVBB is not expected to discriminate efficiently these ones.

The second experiment (Tables 6.5 and 6.6) shows that in the worst case there are 162/300 candidate instances. This means that half of the candidates have been discarded. In the best case, only five candidates remain out of 300. For the classes, in the worst case, 19 classes are discarded (32 remaining) and in the best case only one possible class remains. In this last case, the recognition is completed only with MVBBs and context. Moreover, the results show that the color MVBB is discriminative at instance level. However at class level they discard few models. This is due to the dataset construction based on semantics and no objective criterion. Indeed, different color objects can belong to a same semantic class, yielding a color bounding box with little discriminative power. Regarding the context, it filters out some classes removing large quantities of candidate instances, up to 53 instances. In some cases, like the keyboard class, the MVBBs have already singled out this class so the context does not bring any additional information.

The last experiment (Figure 6.13) shows that a simple cascade of weak descriptors is not sufficient to obtain good classification performances. Most of the classes perform poorly because

Figure 6.12: A color modified by the highest and lowest standard deviation. From left to right: Lab = (75, 75, 75), (72, 71, 71), (48, 50, 55). There is little difference between the first and second.

Table 6.5: First line: remaining instances after spatial MVBB; second line: remaining instances after spatial+color MVBBs; third line: remaning instances after spatial+color MVBBs + place as context. For each confidence level, the two test examples with lowest and highest number of candidate instances are represented.

Confidence	**90%**		**95%**		**99%**	
Lowest						
Spatial	5	46	5	31	12	31
Spatial+Color	5	9	5	18	12	31
Spatial+Color +Context	5	9	5	18	9	22
Highest						
Spatial	131	136	158	159	205	209
Spatial+Color	120	127	147	151	200	203
Spatial+Color +Context	80	105	98	114	147	162

of a high false alarm rate, though it is the desired behaviour. However, the curve for the keyboard class exhibits good performances. This is coherent with the last experiment where the keyboard class manages to retain only 1 candidate. Moreover this result can be explained by the fact that keyboards have larger dimensions than the rest of the objects in the dataset.

This section puts forward the use of weak descriptors and contextual information to efficiently match an unknown object against a database of models. In order to solve this problem a cascade of weak descriptors is proposed. The results show that using the cascade allows discarding an important part of known models. Finally, we showed that a simple cascade can be used as an object classifier when some objects have outstanding dimensions or color in the database.

6.3 Conclusion

Through this Chapter, we have proposed a solution to handle the object's place as a contextual source by first learning it and then using it to increase recognition speed. The first section presented a method to autonomously explore, model, segment, find the areas-places and object-areas links. The results showed that despite a noise inherent to modelling with a robot using a low cost sensor, the segmentation and classification stages can be performed correctly, yielding the areas-places links. Then linking objects and areas can be done over time by the robot. Though the experiment has been performed with research grade software it yielded interesting results. The sources of error are analysed and are mainly attributed to noise during the modelling step

Table 6.6: First line : remaining classes after spatial MVBB; second line : remaining instances after spatial+color MVBBs; third line: remaning instances after spatial+color MVBBs + place as context. For each confidence level, the two test examples with lowest and highest number of candidate classes are represented.

Confidence	**90%**		**95%**		**99%**	
Lowest						
Spatial	1	4	2	6	5	12
Spatial+Color	1	4	2	6	5	12
Spatial+Color +Context	1	4	2	6	4	10
Highest						
Spatial	37	39	41	41	47	47
Spatial+Color	36	38	40	40	46	47
Spatial+Color +Context	22	25	24	26	29	31

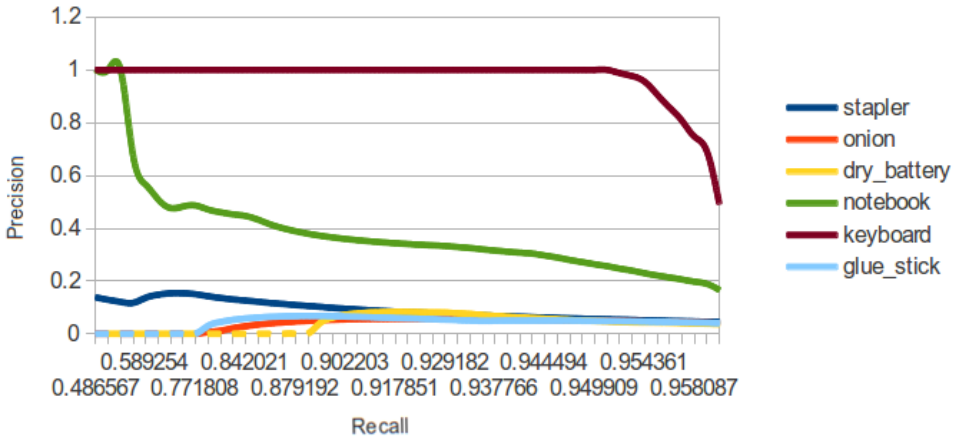


Figure 6.13: Precision-recall curves for six classes representatives of the dataset.

and to recognition errors. Different enhancements are proposed to improve the current results, for example online correction of the offline segmentations.

The second section focuses on showing that the combination of simple visual cues with contextual information allows removing large numbers of candidates when doing object recognition. The list of candidates is filtered through a visual and contextual cascade which removes unlikely candidates. Results show that in the worst case, half of the candidates are removed while in the best case only five candidates remain.

The work presented in this Chapter puts forward the fact that the place where objects are found is an important piece of information and using it can relieve the complexity of recognition tasks. Moreover, learning the object-areas relationships is crucial to answer commands like "fetch OBJECT" where no contextual information is provided by the command. This is of particular importance as current visual methods have a limited viewing range, so knowing where to look for an object is necessary before actually seeing it. The place context is important, but next Chapter shows that detecting other objects can also help recognition and provide cues as to where some

objects can be found.

Chapitre 7

Ce second Chapitre dédié au contexte pour la reconnaissance est consacré au relations contextuelles objet-objets. En effet, certains objets tendent à apparaître fréquemment ensembles, par exemple un chaussure avec sa paire ou une chaise avec une table, ils sont dit co-occurrants. Cette tendance à apparaître ensembles peut être utilisée pour faciliter la reconnaissance, en effet si un objet est visible il y a des chances que les objets co-occurrents soient aussi dans la scène. A l'inverse, la présence de certains groupes d'objets peut signifier l'abscence d'objets qui ne sont pas co-occurrents avec eux. Nous allons voir dans la suite que apprendre ce type de contexte est difficile, nous allons néanmoins proposer des solutions. Par la suite nous montrons la problematique de l'inférence avec ce contexte et proposons une solution dans la continuité de l'état de l'art.

Pour apprendre le contexte objet-objets, la plus part des méthodes de l'état de l'art utilisent des bases de données annotées. Or, ce type d'information est fortement biaisée de par le choix des images inclues dans la base de données. De plus, la création d'une telle base de données demande un travaille colossale pouvant prendre plusieurs mois à plusieurs centaines de personnes. La seule autre solution proposée jusqu'à maintenant pour apprendre le context objet-objets était d'utiliser l'outil Google Sets. Cette outil répertorie les listes de noms à travers le web et associe les noms qui sont souvent cités dans une même liste. Cette méthode permet d'obtenir de manière automatique et rapide un contexte objet-objets. Malheureusement, le service Google Sets n'existe plus.

Afin de combler ce manque de méthodes automatiques, nous proposons d'utiliser les données d'Amazon.com pour estimer la co-occurrence entre des objets. En effet, sur Amazon.com, lorsqu'on observe un produit, une serie d'autres produits sont proposés. Cette aide s'appelle les « Produit Achetés Frequement Ensembles » (PAFE). En supposant que des produit achetés ensembles apparaissent ensembles dans le monde on peut obtenir une information de co-occurrence depuis Amazon.com. Par ailleurs, en utilisant le rang des ventes pour chaque produit, il est possible de classer les produits qui apparaissent le plus souvent avec un objet donné. Ceci permet non seulement de connaître les objets co-occurrents, mais également d'avoir une probabilité de co-occurrence.

Pour ce faire nous passons par l'API Amazon.com qui permet d'obtenir ce genre de renseignements. Nous construisons trois matrices qui combinent les informations citées ci-dessus : la première est binaire et indique juste la co-occurrence, la deuxième contient la frèquence avec laquelle un objet apparaît, la dernière prendre en compte le classement des ventes. Des exemples des trois matrices sont fait pour dix objets d'electronique et dix objets de la cuisine.

Les trois matrices présentent des résultats cohérents entre elles et avec le bon sens. Néanmoins, à cause du fonctionnement du PAFE, les objets ont une forte probabilité de co-occurrence avec eux mêmes. Par ailleurs, certaines classes dont les noms sont proches, comme « verre à vin » et « verre » provoquent une quantité faible de résultats pour la classes la plus specialisée, « verre à vin » dans cet exemple.

Les avantages d'une telle méthode sont qu'elle permet d'obtenir des informations sur la co-occurrence d'objets de manière automatique et presque immédiate. Etant donné l'existance de plusieurs sites amazon dans plusieurs pays, cette information est également adaptée à différentes cultures. Les limitations de cette approche viennent de l'opacité du fonctionnement de l'outil PAFE, du faible nombre d'objets présent sur Amazon.com par rapport au monde réel et du fait que ces site ne reflète les habitude que d'une partie limitée de la population. Il s'agit néanmoins de la seule alternative en dehors des base de données annotées et nous pensons que en utilisant cette approche pour créer un base de données de départ, un apprentissage en ligne peut par la suite continuer d'enrichir les connaissances du robot avec des données adaptées à son environnement

particulier.

A l'aide des matrices de co-occurrence obtenues précédement, nous allons tenter de rendre plus rapide et plus robustes la reconnaissance d'objets. La principale difficulté dans la reconnaissance d'objets avec des objets co-occurrents, c'est qu'il faut déjà avoir un ou plusieurs objets qui donnent de l'information pour identifier les autres. Alternativement, les méthodes de l'état de l'art se sont tourné vers une solution qui consiste à trouver la combinaisons la plus problable d'identitées pour les objets visibles en prenant en compte les descripteurs visuels et les contraintes contextuelles.

Pour ce faire, la majorité des méthodes utilisent des techniques de graphes probabilistes, tels que des Reseaux de Markov ou des Champ Aléatoires Conditionels. Ce méthodes ont donné de bons résultats mais il existe autant de manière de construire de tels graphes que de travaux. De plus la définition de ce type de graphes demande une expertise en graphes probabilistes et dans le problème en question. Nous proposons de simplifier et formaliser cela en utilisant une méthodologie apparue récemment : les Reseaux de Markov Logiques. Ces approches permettent de transformer un ensemble de contraintes, sous la forme de formules logiques du premier ordre, en graphe probabiliste. Par ailleurs, des méthodes d'exploration permettent de trouver les contraintes existantes à partir d'un jeu de données, ce qui permet d'automatiser la rédaction des formules logiques. Les problème au final se résume à proposer un ensemble d'axiomes de base et une base de données qui permettra de les enrichir. Dans le cas de la détection d'objets la situation est particulière car il existe de types de formules : celles qui dénotent d'un fait variables, les résultats de reconnaissance par exemple, et celles qui dénotent de fait fixes, le contexte par exemple qui est connu une fois pour toutes. Nous fournissons un exemple de formules de bases pour le cas de la détection d'objets.

Afin de vérifier le bon fonctionnement des méthodes décrites ci-dessus nous mettons au points une expérience sur des données réelles. Le choix des données est important dans l'évaluation d'une méthode basée sur le contexte. En effet, si les images de la base de teste n'ont pas de contexte, l'information contextuelle ne pourras pas être utilisée. Or c'est le cas de beaucoup de bases de données dont les images montrent des objets seuls. Dans notre cas, nous souhaitons disposer de plusieurs objets connus visibles dans la scène. Très précisément pour combler ce manque, des chercheurs ont développé la base de données COCO qui possède de nombreuses classes apparaissant ensembles dans les images. Nous utilisons donc cette base de données puisqu'elle est adaptée à des techniques basées sur le contexte. Pour obtenir une liste de candidats pour différentes régions des images tests, nous prennons un classifieurs d'objets de l'état de l'art.

La base de donnée COCO étant jeune, les serveurs d'évaluation permettant de tester des algorithmes ne sont pas encore disponibles. Nous ne fournissons donc pas de résultats pour ce Chapitre.

Pour conclure ce deuxième Chapitre traitant du context objet-objets, nous avons montré qu'il était possible d'apprendre des données contextuelles de ce type depuis Internet en utilisant les bases de données Amazon. Nous avons montré trois façon d'exploiter ces données pour obtenir toujours plus de précision. Nous avons également montré une approche permettant de générer automatiquement des outils probabiliste aidant à la reconnaissance d'objets et permettant d'exploiter le contexte. Pour finir nous avons définis une expérience permettant d'évaluer les différents éléments de notre pipeline. Dans l'ensemble de ce Chapitre l'accent est mis sur les méthodes automatiques qui permettent de générer des information ou des traitement compliqués à partir d'un ensemble de données simples.

7

Learning and Using Objects as Context

> When you have eliminated all which is impossible, then whatever remains, however improbable, must be the truth.
>
> Sherlock Holmes

Without visual information, but a vague shape, the objects hidden in Figure 7.1 can be easily identified thanks to objects co-occurrence.

Objects co-occurrence is the frequent apparition of given objects together. It stems from the fact that some objects are designed to be used in conjunction and thus frequently appear close to each other. For example, a keyboard, mouse and monitor tend to appear on the same table and likely close to each other. This also includes identical objects appearing in given numbers, like shoes or car wheels. These objects are called co-occurrent objects. Objects co-occurrence can be expressed as the probability for a set of objects to appear close to each other. It also informs on the probability of NOT finding some objects in the surroundings of another one.

As seen in the previous Chapter, this kind of information is valuable when doing object recognition as it allows to significantly reduce the candidates list for an unknown object. It can also detect recognition errors if the object identity is inconsistent with respect to the surrounding objects. In cases such as the one depicted in Figure 7.1, it may come to the point where recognition is accomplished only from co-occurrence information. Co-occurrence provides useful information and complements visual data with a priori knowledge.

In the following we provide a state of the art of object-object context learning. It shows that all available methods learn context from handmade datasets. To provide an alternative we propose to learn the co-occurrence context from the Internet, specifically from Amazon.com. The second part of this Chapter proposes a state of the art of visual inference with contextual information. We show that classical works rely mainly on probabilistic graphical models (PGM), efficient but hard to build. In order to simplify the PGMs creation, we propose to rely on Markov Logic Networks (MLN). From a simple set of first order formulas and actual values, they generate automatically a Markov Network. Finally, we propose to evaluate the previous methods on a dataset. A quick review of the existing datasets brings our attention to the COCO dataset. Because of time constraints, the results of this work are not available. Nevertheless, the methods and experiments are detailed.

Figure 7.1: The hidden areas do not provide visual information. Still, the surrounding objects provide enough clues to identify the hidden ones as plates.

7.1 Learning The Objects-Objects Relationship

In a controlled environment, the objects repartition and their co-occurrence can be easily controlled and computed. However, getting data from the real world is a lot harder as this kind of knowledge comes from experience, from seeing again and again scenes with known objects. To a limited extent, some co-occurrence information can be gained from the Internet thanks to two methods: Google Sets, which has sadly been discontinued, but is nevertheless presented in Section 7.1.1 and Amazon's "Customers who Bought this Item also Bought" feature, hereafter abbreviated to CBIB, as is shown in Section 7.1.2.

Since Google Sets shut down, no equivalent tool has been made available. The contribution of this work is an alternative to Google Sets, to automatically retrieve co-occurrence matrices for sets of objects from Amazon.com. The main hypotheses of this section are that items bought together are likely to be co-occurrent objects and that the more an item is sold, the more it is likely to appear in the world. To put this work in context, previous works in learning object-objects co-occurrence are described hereafter; the focus is on the learning methods.

7.1.1 Previous Works

Multiple works rely on objects co-occurrence to increase recognition performances [12, 28, 133]. In [12], the authors search for out-of-context objects. They use the SUN09 dataset to learn a latent tree. Lin et al. [28] use context to help object recognition with bounding boxes. A loss function representing the context is learnt from the NYU v2 dataset. In the work from Ladicky et al. [133], the authors learn a co-occurrence cost function from the MSRC and Pascal VOC datasets and use it for enhanced scene labelling. As can be seen in Table 7.1, most of these works rely on learning from labelled datasets. However, labelling datasets is a burdensome task and, as noted by Galleguillos et al. [10], when training data is weakly or not labeled, an external source of context is needed. To the best of our knowledge, there is a single work proposing such an external source of context: the work from Rabinovich et al. [134]. In this work, the authors propose to use Google Sets [135] to retrieve a binary co-occurrence matrix for a given list of

Table 7.1: Previous works dataset sources, the data type and what is learned from them.

Work	Source	Type	Output
[12]	SUN09	Labeled RGB	Latent tree
[28]	NYU v2	Labeled RGB-D	Loss function
[133]	MSRC Pascal VOC	Labeled RGB	Cost function
[134]	MSRC Pascal VOC Google sets	Labeled RGB Internet	Binary co-occurence matrix
Ours	Amazon's CBIB feature	Internet	Numerical co-occurrence matrix

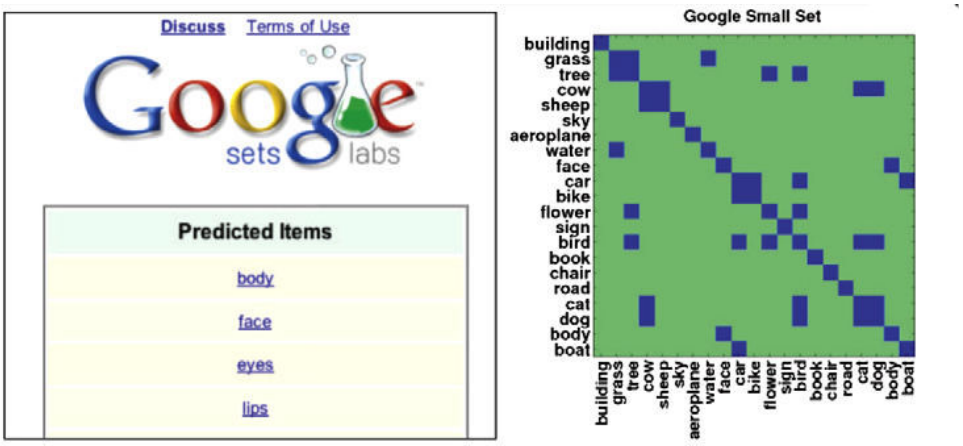


Figure 7.2: The Google Sets homepage (left) and a binary co-occurrence matrix built from its output (right).

objects.

In the following, we briefly describe the workings of Google Sets, its assets and limitations. As the name indicates, Googles Sets crawls the web, searching for lists (sets) of words. Each word is then associated with the most frequent words of these lists. When words are provided to Google Sets it returns a ranked list of the words most frequently associated with the input words. This ranked list can easily be turned into a binary co-occurrence matrix. The hypothesis made in [134] is that objects that co-occur in lists co-occur in images. This solution works for many words, yields impressive results and allow learning co-occurrence information automatically. Nevertheless, it has some limitations, like cow co-occurring more with sheep than with grass, or the fact that the returned matrix is binary. The main problem is that Google discontinued the Google Sets service. Though it is theoretically possible to build a similar service, it is likely to be limited without Google's mass of data and computing power.

In their conclusion, the authors of [134] point to another potential source of contextual information: Amazon.com. Though they envisioned using it to retrieve semantic object hierarchies, we show in the next section how Amazon.com can be queried to obtain numerical co-occurrence matrices.

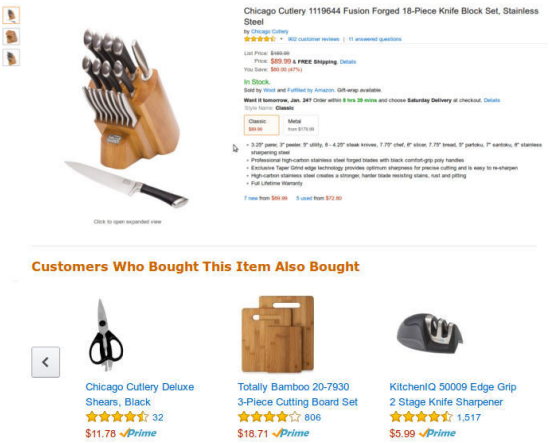


Figure 7.3: The CBIB feature provides a list of items commonly bought with the considered item.

7.1.2 Learning co-occurrence from Amazon.com

As a source of information, Amazon.com has various advantages. With its different versions for different countries, it provides culture dependent data. This is crucial for object co-occurrence, for example in occidental countries plates often co-occur with forks, this may not be true in oriental countries. It provides up-to-date information as the data is permanently updated as customers use the website. This allows taking into account evolution of habits, trends, etc.

To access this data, Amazon.com provides the Product Advertising API. It is used to advertise products on websites. But it can also be used to query Amazon.com for information. In this particular case, two features are queried, the "Customers who Bought this Item also Bought" and "Best Sellers Rank" features. For a given class of objects, the first provides a list of co-occurring objects. The second one allows ranking the co-occurring objects. Both features are described in more detail hereafter.

Customers who Bought this Item also Bought

When looking at an item page, Amazon.com proposes a list of items which have been bought with the current item by previous customers. This is called the "Customers Who Bought this Item also Bought" feature, CBIB for short. The list of proposed items belong to the same department as the considered item.

The hypothesis in this work is that items appearing in the CBIB feature list co-occur with the considered item. One can take advantage of this to create a co-occurrence matrix for a list of objects classes.

Consider a list of classes whose co-occurrence matrix is wanted. For each class, the regular Amazon.com search is queried using the class name, this yields a list of items. For each item, the CBIB feature is queried to retrieve a list of co-occurring items. As only the objects from the list of classes are of interest, the proposed co-occurring items are filtered to keep only those appearing in the list of classes.

This approach provides a binary co-occurrence matrix. To obtain fine-grained information, a matrix based on the frequency of apparition of the products in the CBIB feature is also computed. So we count the number of times a class appears in the CBIB feature for items of a given class.

Amazon.com also provides a ranking of the products sold for a given department, this can be used to retrieve more precise information.

Best Sellers Rank

The best sellers rank feature associates items with a rank. This rank mainly depends on how many items have been sold recently. Items with a zero rank have not been sold for some time, while items with no rank have just been introduced.

The idea when using this feature is that an item which has a high rank is sold in higher quantities than items with lower rank. So it should appear in the world more frequently than other items and thus have a higher co-occurrence weight than these items. Though it provides no absolute information, this weight allows ranking the co-occurrent items for each class.

The most sold items have low rank while little popular items have high rank. To obtain the weight, the rank is inverted so a low rank provides a high weight. For a given class A, the weight of all co-occurrent items belonging to a given class B are summed to obtain the final co-occurrence weight for the couple A-B. This allows building a co-occurrence matrix taking into account the items ranks.

7.1.3 Experiments: Obtaining Object-Objects Contextual Information

The method described previously provides three co-occurrence matrices: a binary matrix, a frequency matrix and a rank matrix (Figures 7.4). The matrices are not symmetric and should be read along the rows.

The co-occurrence information is extracted as explained in Section 7.1.2. In order to be compatible with the versions of Amazon.com available in various countries, no advanced searching/sorting feature, like using sub-departments, from the Product Advertising API is used. For each class of object, N instances of the object are considered. For each instance M items are proposed by the CBIB feature. In this experiment, N = 10 and M = 10, yielding a total of 100 co-occurent candidates. These numbers are enforced by the Amazon API.

Examples of matrices corresponding to the Kitchen and Electronic super categories of the COCO dataset [136] are shown in Figure 7.4.

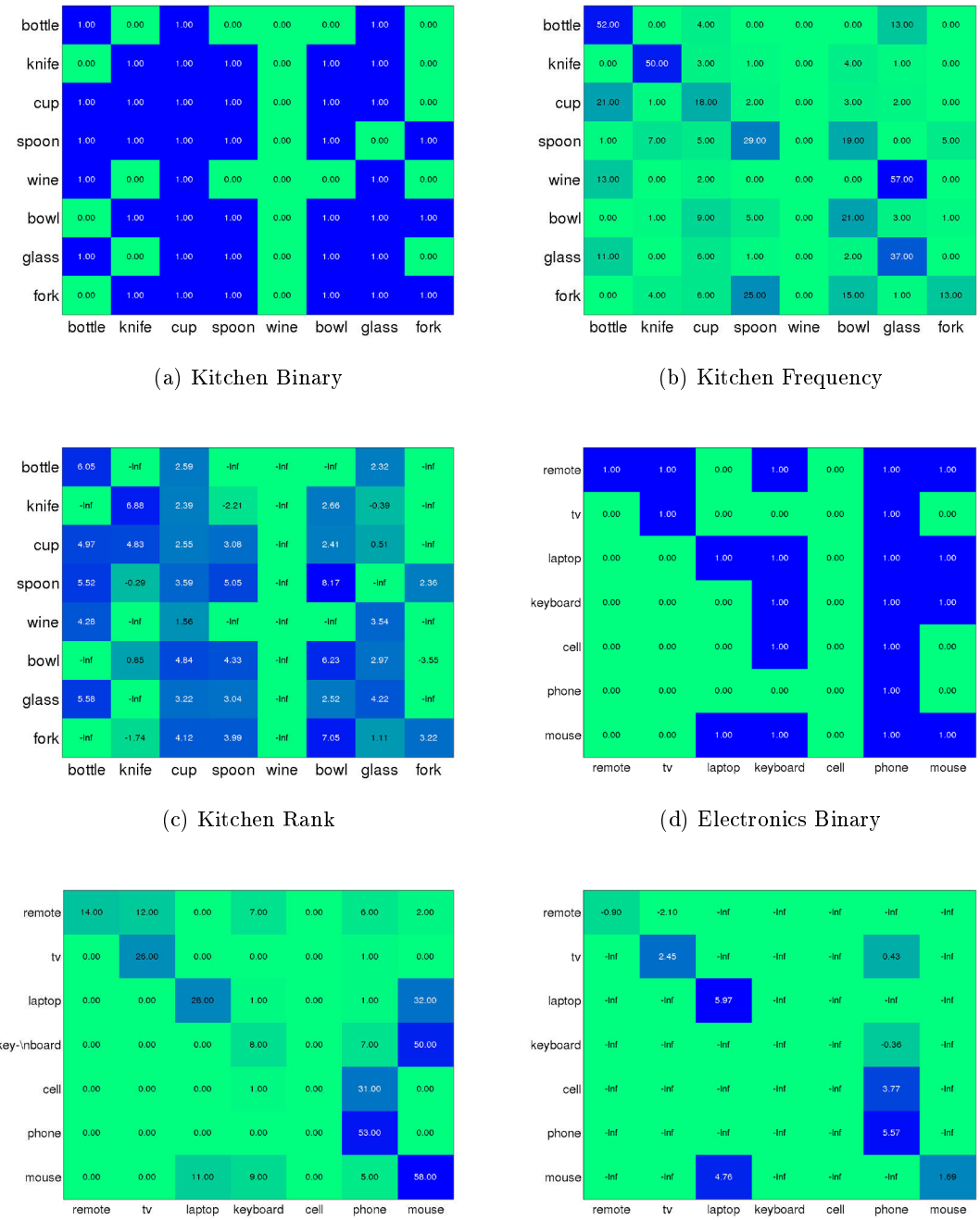
7.1.4 Results

By construction, the resulting matrices are not symmetrical. The results should be read along the rows, not along the columns. For the rank matrices, a value of $-Inf$ means that there is no co-occurrence between the objects.

From a global point of view, we can see that the matrices have a similar structure, which shows that the three methods have coherent results.

One can note that, with a few exceptions, objects tend to co-occur with themselves. This is due to the CBIB feature over-representing similar items in its recommendations. This bias is less notable in the rank matrices. The particular cases are the *wine* (wine glass) and *cell* (cellphone) categories. They are special in the fact that they are semantically already represented with a more common name, *cell* and *wine* are respectively included in the names *phone* and *glass*. We believe these categories were separated in the COCO database because, though semantically similar, they have different appearances.

Finally, the Electronics Rank matrix has many $-Inf$, this may be due to the products changing fast in this domain, so they have no time to get a ranking.



(a) Kitchen Binary

(b) Kitchen Frequency

(c) Kitchen Rank

(d) Electronics Binary

(e) Electronics Frequency

(f) Electronics Rank

Figure 7.4: Examples of co-occurrence matrices for the Kitchen and Electronics super categories of the COCO dataset. To enhance visualization, because of its large range values, the rank matrices are in fact the log of the rank matrix.

As an example, we comment the results for the fork and mouse categories. In the binary matrix (Figure 7.4a), the fork co-occurs with the knife, cup, spoon, bowl, glass and fork categories. In the frequency matrix (Figure 7.4b), the fork co-occurs with the spoon(25), bowl(15), fork(13), cup(6), knife(4) and glass(1) categories, with the number of co-occurrences in brackets. According to this matrix it is more likely to find a fork near a spoon than near any other object, and the less likely objects to be found near a fork are bottles and wine glasses. According to the rank matrix (Figure 7.4c), the fork co-occurs with the bowl(7.05), cup(4.12), spoon(3.99), fork(3.22) and glass(1.11) categories, with the category rank in brackets. The ranking is different from the one provided by the frequency matrix as the more likely object co-occurring with a fork is a bowl, the spoon being in third position.

In the mouse case, the mouse category co-occurs with the laptop, keyboard, phone and mouse categories according to the binary matrix; with mouse(50), laptop (11), keyboard(8), phone(5) categories according to the frequency matrix; with laptop(4.75) and mouse(1.69) according to the rank matrix.

In both cases, the co-occurent objects and their ranking seem reasonable. Though the frequency and ranking matrices provide different results, we could not come up with an experimental approach to determine which provides the best information.

7.1.5 Limitations and Benefits

The main limitation comes from the fact that this method is useful for objects present, and widely bought, on Amazon.com. To the contrary of Google Sets, it does not provide results for any category of object. Another limitation comes from the mechanisms used by Amazon.com to select the items shown in the CBIB feature. It tends to over-represent items of the same class as the considered item. Moreover, for departments such as Grocery, the CBIB feature provides poor propositions.

On the other hand, Amazon.com represents an always evolving and up-to-date source of information. It is one of the few sources adapted to various cultures. And it may be a better approximation of the real world frequencies than Google Sets. The results showed in this work are sensible and though not perfect, they can form a solid basis to be further enriched by online learning. In Section 7.3, an experiment is proposed to evaluate the quality of the matrices obtained with this method and choose the most appropriate one.

7.2 Inference knowing Objects-Objects Relationship

Once contextual information has been learned, it can be used to enhance a visual object categorisation task. The idea is to use a multi-class categorisation algorithm which yields, for various regions of the input image, a ranked list of candidates with associated probabilities. This list can then be re-ranked thanks to the contextual information, to obtain the most likely combination of objects in the scene. However, combining interdependent visual and contextual cues requires complex probabilistic models.

7.2.1 Previous Works

Historically, in the nineties, the active vision community has tackled these kind of intricate problems using Bayesian Networks related methods [140, 141].

In recent literature, some works focus on simple approaches. In [137], the authors identify an unknown object from its distance to a known object. Using a naive Bayes approach they

Table 7.2: State of the art in context and vision fusion for categorisation.

Ref.	Model	Learning	Inference
[137]	Naive Bayes	Naive Bayes	Naive Bayes
[138]	Desai scores	SVM	Branch and Bound
[139]	MRF	SVM	MIP/QPBO
[134]	CRF	Monte Carlo Gradient descent	Bayes-like
[28]	CRF	Primal-Dual	Maximum A Posteriori
Us	MLN	Done a priori	MaxWalkSat

combine manually labelled objects and their euclidean distance to the object of interest. This work is limited to pairwise relationships. For multiple relationships, Southey et al [138] use a Desai Score (DS) to merge a set of labels with a set of distances. These approaches can handle very basic cases closer to experimental setups than to real situations.

For complex cases involving multiple visual and contextual features, naive methods are not sufficient, recently most works turn to graphical models. Graphical models are powerful methods but require specific designs to avoid high complexity plus learning and inference methods adapted to this design.

In [139], the authors represent the relationships between various features and contextual cues with a Markov Random Field. To increase the discriminative power without adding too much complexity, they represent some relationships as associative edges, more powerful but with more parameters, while other relationships are modelled with non-associative edges. The edge's potential parameters are learned with a Support Vector Machine (SVM). The inference problem is solved with a mixed-integer program and they show that, by relaxing some constraints, the problem can be solved with Quadratic Pseudo-Boolean Optimization.

Rabinovitch et al [134] use a Conditional Random Field (CRF) to model undirected dependencies. The CRF is fully connected to model multiple object dependencies. For complexity reasons, the learning is made on simple cases. The partition function is learned with a Monte Carlo Integration and a gradient descent to find the potentials. Inference is done through Bayes-like relationships.

Lin et al. propose in [28] to use a CRF with four different potentials: a unary potential for the scene label, a unary potential for the object label, a binary potential for the scene-object context and a binary potential for the object-object context. The weights of the model are learned through a primal dual framework. Inference is done through a Maximum-A-Posteriori approach.

These works rely on Markov or Conditional random fields, respectively directed or undirected graphs, to model the problem. However, each work defines its own potentials and edges organisation. To the best of our knowledge there is no work in this field using a unified framework to include the visual and contextual relationships.

7.2.2 Markov Logic Networks

In order to formalise and automate probabilist graphs creation, Richardson and Domingos proposed in [142] a mathematical framework which allows constructing Markov Network (MN) from first order logics sentences with associated weights: the Markov Logic Networks (MLN).

Table 7.3: Example of weighted formulae forming a Markov Logic Network.

0.7	$Actor(A) \Rightarrow \neg Director(A)$
1.2	$Director(A) \Rightarrow \neg WorkedFor(A, B)$
1.4	$Movie(T, A) \wedge WorkedFor(A, B) \Rightarrow Movie(T, B)$

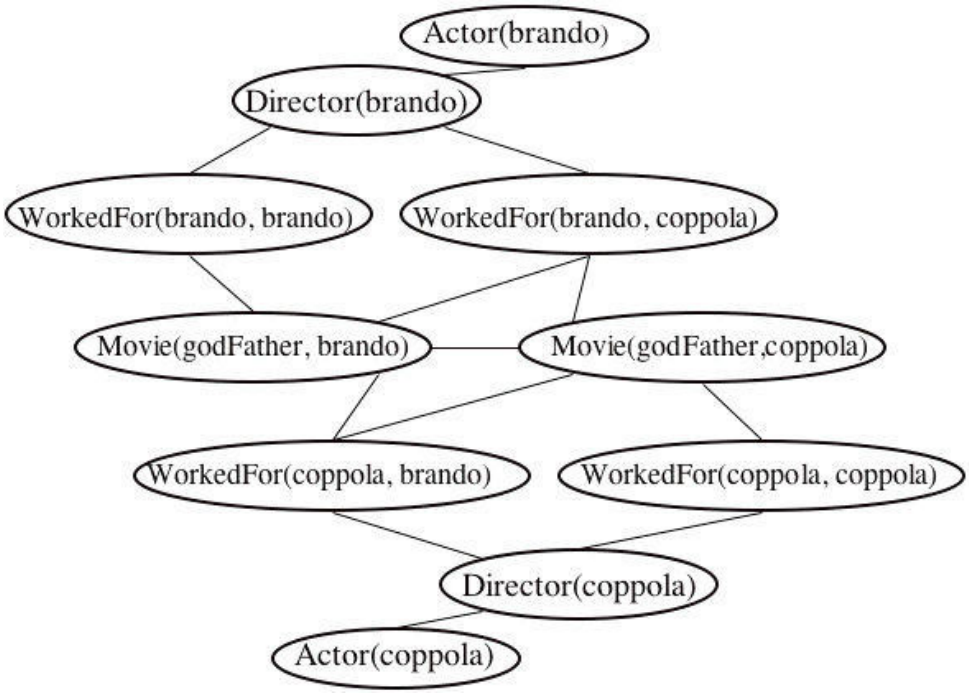


Figure 7.5: An example of grounded Markov Network obtained with the Markov Logic Network from Table 7.3. The constants are brando,copolla and godFather.

A MLN is a collection of first order logic formulae with associated weights (Table 7.3), these are used as templates to create features of a Markov Network (Figure 7.5). When truth values are associated to the logic formulae, i.e. when the formulae are grounded, the MLN becomes a grounded MN. Each grounded formula becomes a vertex of the grounded MN and vertices are joined by an edge if and only if there is a relationship between them. The MLNs have numerous advantages. They are easy to design, indeed it is easier for non-experts to design logical sentences than graph potentials. As shown later, even if the design is flawed, learning methods can correct it automatically. And the graph is as connected as needed, a relationship exists in the model only if it exists in the observed data.

Various works showed the usefulness of such approach. In [143], the authors compare a MLN with state of the art Collective Classification (CC) algorithms. They find that the MLN lags behind the classical CC algorithms for simple datasets, but it outperforms them for datasets with complex relationships. The authors of [144] use a MLN for word spotting in ancient documents and show a 50% increase in precision compared to standard methods.

However, the previous works assume static data. For categorisation, the incoming data changes while the contextual information remains the same. Such cases have been specifically treated in the literature. In [145], MLN are used for face detection. They enforce simple rules like "a same face cannot appear two times". In [146], a MLN is used to find suspicious naval behaviours by combining naval rules dependent on the type of vessel and their trajectory. Fi-

Table 7.4: Symbols used to build a MLN for object categorisation with vision and context.

Constants	$o1...N, l1...N$
Variables	$Objects, Labels$
Functions	$Label(o, l)$ $Cooccurr(l1, l2)$
Predicates	$label(o1, l1) \wedge Cooccurr(l1, l2) \Rightarrow lalbel(o2, l2)$

nally, Song et al. [147] use a MLN to combine known spatial object-object relationships with action recognition algorithm outputs. The general idea is to separate MLN's formulae in two families. The "Given" predicates, which represent a priori information, in our case the contextual information, and the "Observed" predicates, which match data received at inference time, in this case the visual data. The advantage of this situation is that the predicates used as query and those used as evidence are known in advance. As shown hereafter, this facilitate the learning and inference steps.

Learning

When a dataset is available, learning methods allow constructing weighted formulae from the data. The formulae with associated weights are called the structure in the following.

Learning has been first addressed in [148] and later refined in [149]. The authors propose a generative method to learn the structure by maximizing the likelihood with a Quasi Newton method (L-BFGS). With fixed weights, a search in the space of possible formulae allows adding the best, or k bests, formulae at a time. The best is defined as the one providing the maximum likelihood. To learn the weights, a discriminative approach is presented in [150]. The learning maximizes the conditional likelihood of query predicates given the evidence ones. This is done through a voted perceptron approach. If a set of formulae is already available, the aforementioned methods allow correcting and refining the formulae on top of finding the weights for each formula. In any case, learning algorithms perform better when at least the unary relationships are provided as a start [149]. For large datasets, the learning can be complex so some works rely on a hand made structure.

Inference

Given a MLN and input data, the MLN contains weights and formulae while the data provides constant values. These constants are used to ground evidence formulae which allows building the grounded MN. To build the MN, the most likely state of queries given a set of evidence is found through a solver. Various inference solvers are compared in [150]. It appears that the MaxWalkSat solver performs the best.

7.2.3 Modelling Object-Object Co-occurrence

In this work, we propose to use a MLN to merge information about objects co-occurrence and visual categorisation. The co-occurrence is known a priori through learning. For an image with a set of objects, the visual categorisation provides, for each object, a set of candidates labels with associated probabilities.

To use the MLN approach, one needs to design a set of formulae describing the various information at hand and their relationships. The formulae are constructed out of four types

of symbols: constants, variables, functions mapping objects to object and predicates describing relations among objects. The symbols used in this case are summed-up in Table 7.4. In this case, the "Given" predicates are made of the single function $Cooccurr(l1, l2)$ which denotes co-occurring categories. The "Observed" predicates are made of the function $Label(o, l)$. The visual categorisation method returns various possible labels for a single object, so a set of $Label(o, l)$ predicates with varying l and weight.

7.2.4 Limitations and Benefits

The Markov Logic Network is a principled way to build graphical models from data alone or from a set of first order logic formulae. It allows integrating different contextual and visual sources as long as the relationships between them can be expressed with first order logic. Though, care must be taken to keep the model simple as this framework can struggle with large quantities of data. As will be shown in the following section, we avoid this pitfall by providing hand made logical formulae and providing the weights associated with each formula. With the learning problem solved, inference is done on images only. So the inference problem is also kept simple as no more than a few objects of interest are visible at the same time. The resulting graphs are simple and their connections are specific to the data at hand.

7.3 Experiment: Combining Amazon's Context and Visual Information with a MLN

In this section, an experiment is presented to evaluate the efficiency of a Markov Logic Network in merging contextual and visual information. It also allows estimating the quality of the information provided by the matrices created from Amazon.com. The object-objects context is extracted as explained in Section 7.1. The visual labelling is done with Felzenszwalb's categorisation algorithm [151]. Both outputs are merged using a Markov Logic Network, as explained in Section 7.2 and the resulting pipeline is tested on the COCO dataset presented hereafter.

7.3.1 Dataset

This work aims at evaluating a pipeline based on visual and contextual information, where objects form the context. In order to validate such method, one should use a dataset with images where co-occurrent objects appear together. Moreover, the objects should be found on Amazon.com; these tend to be small everyday life objects.

However, most datasets are built on images centred on objects without much context [152], [153], [154]. Moreover, these datasets have an unbalanced number of images per category and usually small objects are not numerous. For example, the "computer mouse" category has 90 images in the SUN dataset [112], 47 images in the Caltech-256 dataset and 1303 images in ImageNet(3.0) [154].

The new COCO dataset [136] offers a good alternative. In the COCO dataset, the images have an average of 7.7 instances per image which represent a correct amount of contextual information. Every class has around 10,000 instances in the dataset, for example the "computer mouse" category has almost 8,000 instances. This allows training efficient detectors as is explained hereafter. Finally, according to [136], this dataset is harder than the classical PASCAL challenge dataset, thus increasing the need for contextual information

This work considers only objects that can be found on Amazon.com, for this reasons we consider the following super-categories : accessory, kitchenware, appliance, electronics, indoor

objects. Though the sports super-category objects can be found in Amazon.com, we believe these objects rarely co-occur, except for the ball category with all the others, so there is little contextual information to use in this super-category.

7.3.2 Object Categorisation

In the same way as in [136], the initial visual object detection is done with Felzenszwalb's Discriminative Part-based Models (DPM) detector [151]. The DPM allows detecting and localising generic categories in static images.

This method relies on representing object as a set of parts which locations are not known but arranged according to a given configuration (e.g. star configuration = one root + multiple parts).

A sliding window approach is used match portions of the input image with a known object model. The model is composed of templates from the root and the various parts. A given object can be represented as a mixture of models to take into account highly different appearances.

Models are learned with a latent SVM approach on images where the object of interest is labelled with a bounding box. From these bounding boxes, Histogram of Gradients (HoG) features are extracted and a Principal Component Analysis selects the most discriminative features.

One detector is trained for each of the categories mentioned previously with the default parameters. For a given image, the detector outputs a set of regions with a list of candidates for each region. This is fed to the MLN as grounded predicates.

7.3.3 Results: to be continued...

Sadly the COCO dataset team is late on schedule and, at the time of writing, the validation servers for the detection challenge are not up yet. We believe they will come up later in the summer but we will have no time to obtain new results.

7.4 Conclusion

This Chapter has stressed the importance of the objects co-currence as a source of information and proposed two original solutions to the learning and inferences problems. It provided a detailed state of the art in data sources for learning co-occurrence, in classification using objects co-occurrence and in the datasets available to validate such methods. After showing that the shut down of Google Sets left no solution to learn objects co-occurrence from the Internet, we proposed an automated solution based on extracting objects co-occurrence data through Amazon.com services.

To merge this contextual information with visual data, the Markov Logic Networks are presented as an adequate approach. It allows easily integrating contextual data learned through Amazon.com by expressing it as weighted first order logic formulae. The visual data can be integrated with an additional formula, so when constant values are available, the whole Markov Logic Network can be grounded to yield a Markov Network. Finally, an adequate dataset is presented to experiment the combination of these methods. Though results are not available, this framework seems promising.

Chapitre 8

Ce dernier Chapitre met en perspective le travaille précédent en montrant le rôle d'un système de reconnaissance et localisation dans un système robotique complet. En effet, la majorité du travaille présenté se concentre sur une application bien précise, ce Chapitre permet de dézoomer pour voir une application possible d'un tel système.

L'application présente est un système de cobotique pour l'assemblage d'un joint automobile : le joint rzeppa. Il s'agit d'un roulement à bille permettant de transmettre le mouvement de rotation du moteur aux roues même si leurs axes ne sont pas alignés. Afin d'assembler cette pièce, des billes doivent être insérée en force à l'intérieur du roulement. Par ailleurs, les fusées dans lesquelles sont insérés les billes sont lourdes à manipuler. Ces tâches sont pénible et provoquent des contraintes fortes sur les doigts et poignets des opérateurs, débouchant à long terme sur des Troubles Musculo-Squeletiques (TMS). Au delà de l'aspect humain, ces TMS ont un coût pour l'entreprise qui doit les prendre en charge.

La solution proposée dans ce Chapitre est d'éffectuer la tâche avec une collaboration robot/home. Il s'agit d'utiliser un bras robot pour prendre en charge les parties pénibles de cette tâche d'assemblage : la manipulation de la fusée et l'insertion des billes. L'humain s'occupe des parties demandant de la dextérité : l'insertion des billes. Cette répartition des tâches permet de soulager l'humain et de réduire la pénibilité du poste. Les tâches à effectuer sont les suivantes et sont reparties comme suite : prendre la fusée (robot), l'orienter (robot), verifier les défauts (humain), placer la fusee dans l'outil (robot), ouvrir un emplacement de bille (robot), placer une bille (humain), insérer la bille (robot), passer à la l'emplacement suivant (robot), vérifier que le joint tourne bien (robot), placer le joint sur un convoyeur d'evacuation (robot).

Dans le reste de ce Chapitre nous décrivons de manière générale une architecture robotique et utilisons l'application ci-dessus pour illustrer les différents éléments nécessaires au bon fonctionnement d'un robot.

On commence par parler des modèles nécessaire à un robot pour se connaître lui même et son environnement. Tout d'abord le modèle cinematique, qui décrit la façon dont son connectés les différents joints d'un robot (type de liaisons, positions relatives) ; puis le modèle dynamique, qui contient les valeur nominales et maximales de plusieurs paramètres nécessaires aux mouvement du robot ; viens ensuite le modèle inertiel, qui permet de calculer les forces exercées sur le robot et, par exemple, de vérifier que les valeurs données par les différents capteurs sont en accord avec la réalité ; puis le modèle de collision, qui est utilisé pour voir si le robot va rentre en collision avec des parties de l'environnement qui ont également un modèle de collision ; et finalement le modèle visuel, qui permet d'afficher le robot à l'humain mais aussi qui permet au robot de connaître son apparence extérieure. Pour ce qui est du modèle du monde, il est séparé en une partie statique qui est apprise ou fournie une fois pour toute et une partie dynamique qui est mise à jour en temps réel. Ces modèles sont essentiels aux bon fonctionnement d'un système robotique. Nous voyons par la suite les différentes parties du robot et comment elles font usage de ces modèles.

Nous nous plaçons ici d'un point de vue informatique, nous allons donc regrouper tout ce qui est materiel sous le terme hardware. En réalité, le robot est constitué au niveau mécanique de moteurs, transmetteurs, coque, capteurs et au niveau électronique de cartes mères, cartes entrées/sorties, cartes de commande, carte d'alimentation, etc. Tout cela est regroupé sous le terme hardware. Nous continuous au niveau informatique. La couche suivante est le contrôle des moteurs, où les calcules permettant d'envoyer les bonnes tensions aux différents moteurs en fonction de la commande sont faits. La couche suivante est en général une couche d'abstraction qui permet de rendre le reste de l'architecture indépendant du hardware et des logiciels bas

niveau.

Puis viens le générateur de trajectoire qui permet, à partir d'une liste de points, de générer les points que doit atteindre le système à des temps donnés. Cette partie fais un grand usage du modèle dynamique pour ses calculs. Puis viens la gestion des position, cette partie utilise le modèle cinematique pour calculer la position des différentes parties du robot en fonction du mouvement des autres parties du robot. Un autre élément essentiel dans la robotique moderne, c'est le détecteur de collisions et l'environnement de collision. Le monde et le robot sont représentés dans l'environnement de collision et le détecteur de collision s'assure à chaque instant que le robot ne rentre pas en collision avec quoi que ce soit. Cette partie de l'architecture se base sur le modèle du monde statique et dynamique ainsi que sur le modèle de collision du robot. Au dessus viens le planificateur, il calcule des trajectoires 2-D ou 3-D pour le mouvement du robot ; ces trajectoirs doivent éviter les collisions, c'est pour quoi le planificateur communique beaucoup avec le détecteur de collisions.

A ce niveau là se situe une deuxième couche d'abstraction, qui permet de séparer les fonctions vitales du robot des fonctions de raisonnement de haut niveau. Ces dernières sont de deux genre : la supervision et l'interface utilisateur. La supervision choisie les tâches que le robot doit effectuer en fonction des données acquise par ses capteur et de la tâche assignée au robot. L'interface utilisateur permet à l'utilisateur de voir ce qui se passe à l'intérieur du robot et de communiquer avec lui pour lui donner des tâches ou des compléments d'information.

Toutes ces tâches sont ont été implémentées dans le système mis en place dans ce travaille. Les résultat d'un tels système n'étant pas quantifiable, nous avons proposé un ensemble de leçons à retenir de ce projet.

Tout d'abord, lors d'une intégration, il faut partir d'une architecture saine. Il est donc préférable de mettre en place l'ensemble du système avec des éléments génériques et ensuite intégrer les contributions des différents partenaires au fur et à mesure.

Dans ce projet, la pince du bras robot est une pince pneumatique. Ceci implique un certain nombre de précautions, en rapport avec les tubes qui amenent l'air depuis le compresseur jusqu'à la pince. En effet, les mouvements doivent être plannifiés pour ne pas arracher les tubes et la perception doit soit intégrer les tube soit ne pas les voir.

Dans ce projet on a donné beaucoup de réactivité au robot pour qu'il puisse s'adpater aux changement qu'implique un humain dans son plan de travaille. Mais avoir des réflexes statiques peut aussi aider le robot a faire des mouvements repetitif et fiables, ainsi qu'à réduire les calculs lorsque la réactivité n'est pas nécessaire.

Nous avons également constaté qu'une grande partie de l'information du modèle statique et cinematique du robot doit être calculée à la main. Cela prend du temps et provoque des erreurs. Investir dans des méthodes de calibration automatiques permettrait de réduire les deux.

Un autre point est que la vision a tendance à échouer quand placée dans de nouvelles conditions, c'est donc la première chose à tester lors d'un déplacement du setup pour avoir le plus de temps possible pour l'arranger.

Pour finir nous mettons l'accent sur l'utilité de faire plusieurs version du système de plus en plus complexes et intégrant de plus en plus de fonctionnalités. Si une des version ne marche plus, il est toujours possible de retourner à une version précédente.

Il est également essentiel de planifier les accidents possibles et de les prendre en charge à trois niveaux : la préventions, pour empêcher que les accidents arrivent ; l'amortissement, pour réduire la gravité de l'accident quand il arrive ; et enfin, la résolution, pour pouvoir résoudre un problème au plus vite après qu'il ai eu lieu.

Nous concluons ce Chapitre en mettant l'accent sur la complexité d'une architecture robotique et la difficulté que cela représente de la mettre en place. Cela est directement lié à la difficulté

d'intégrer des solutions robotisées dans des milieux industriels. La simplification de cette mise en place peut mener à une adoption plus facile des solutions robotiques dans l'industrie.

8

Industrial Robotics: Cobot For Ball Bearing Assembling

> To succeed, planning alone is insufficient.
> One must improvise as well.
>
> Isaac Asimov

As seen in the previous chapters, service robotics present many challenges due to the high number of parameters: environment, objects, cultures, etc. In such cases, adaptivity through modelling and context learning is of crucial importance. In industrial robotics, the setups are known and standardised. The object's models may be available and there is not much context or it is already known. In such situations, the previously described methods may not be essential. Though, the industry is living a small revolution. The new generation of robots do not just rely on predefined plans, they are capable of collaborating with humans through adaptability and reactivity. Objects modelling, recognition and localisation are important bricks to enable these behaviours.

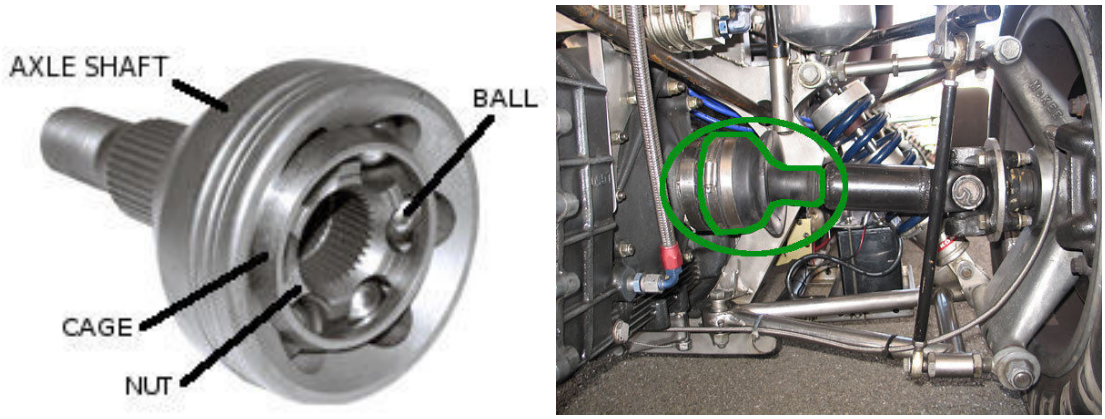
8.1 Industrial Robotics

This Thesis have been made possible through funding from two industrial projects, namely the CAAMVIS (2 years funding) and ICARO projects (1 year funding).

The CAAMVIS project treated of Non Destructive Control (NDC) of large flight parts. The part was modelled at processing time to allow the robot to localise with respect with the part. This in turn allowed scanning precisely the part and localising in space any defect.

The ICARO project involved two human/robot collaborative assembly scenarios. In the first one, a robot and a human need to install rivets between two flight parts. The human inserts the rivets in a grid of holes and the robot localises and installs it. The second scenario requires a robot to ease a ball bearing assembly task so the human performs only the dexterous tasks. The robot needs to localise and manipulate the heavy parts of the ball bearing while the human performs the balls insertion.

In both projects, modelling and localisation are key elements of the process. This is illustrated in the rest of this Chapter through the second ICARO scenario: the ball bearing collaborative assembly.



(a) Perspective view of a Rzeppa joint (b) The Rzeppa joint in a car

Figure 8.1: The Rzeppa joint.

(a) Place the spindle (b) Insert assembled cage + nut (c) Insert balls

Figure 8.2: Rzeppa joint assembling

While the automotive industry heavily relies on robots, tasks requiring high levels of dexterity are still handmade, as is the case for the Rzeppa joint assembling. Such manual assembly task imposes an important stress on the operator's hands, which can lead to injuries in the long term. To tackle this problem, the ICARO project proposes a cobotic solution to automate the toughest parts of the assembly process. The problem and proposed solution are described hereafter in Section 8.2.

The work described in this chapter is the result of a collaboration between Airbus-Group, PSA, Siemens, Tecnalia, LIRMM-CNRS, CNAM, LAAS-CNRS. This thesis contribution to the project is the testing and integration of the partner's contributions into a unified hardware and software architecture which is described in Section 8.4.

8.2 Homokinetic Rzeppa Joint Assembling

The homokinetic Rzeppa joint allows a drive shaft to transmit power through a variable angle, at constant rotational speed. In short, it is a ball bearing joining the wheel and the drive shaft, cf. Figure 8.1. This joint has six balls to ensure fluid rotation. Due to its particular shape, the balls need to be inserted manually during the assembly process.

To assemble the joint, an operator places a spindle on its worktable (Figure 8.2a). A visual and tactile quality check is done on the inside part of the spindle. Then, the operator assembles

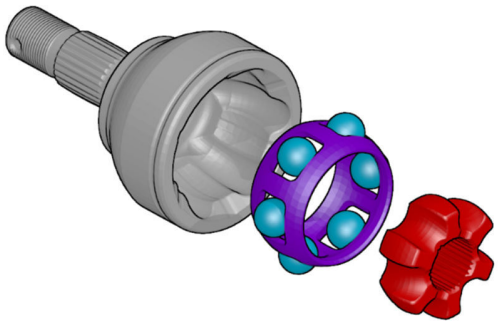


Figure 8.3: From left to right: the spindle, the cage with balls inserted in its sockets and the nut. The cage and nut are assembled and inserted into the spindle, then the balls are inserted. Though this illustration shows the cage with the balls inserted, the balls are the last element assembled.

a cage and a nut and inserts them in the spindle (Figure 8.2b). The operator uses a tool to open in turn each of the cage's socket and insert a ball inside it (Figure 8.2c). Finally, the operator checks that the cage and nut can rotate freely. The resulting joint is shown in Figure 8.3.

The problem stems from the fact that the rotation to open the sockets puts a serious strain on the operator's wrist. Moreover, pushing the balls into the sockets requires a significant amount of effort, especially for the third and sixth ball. Finally, taking and putting away the spindles puts stress on the whole operator's arm. In the long run, these constraints lead to Musculoskeletal Disorders (MSD) for the operators. Besides the human factor, there are economic stakes due to the cost of MSDs treatment.

The ICARO project proposes a cobotic solution to tackle the MSD problems. The spindle recognition, localisation, manipulation as well as the tool motion and parts of the balls insertion are performed by a robotic arm. The operator takes care of the quality checks and place, without effort, the balls in the joint.

The process is illustrated in Figure 8.4. First the system recognises and localises the correct spindle conveyor, as they may be various conveyors with different spindle sizes available. Then the robotic arm picks a spindle from the conveyor. It places the spindle over a plastic part to orient it with respect to the gripper. This is crucial to make sure that the tool will properly enter into the spindle's grooves when inserting the balls. Then, the arm takes back the spindle and shows it to the operator. This one carries out a visual and tactile inspection of the inner part of the spindle. If a defect is detected, the operator tells the robot to throw away the spindle to a scrapheap. Otherwise, he assembles a cage and a nut and inserts them in the spindle. The spindle + cage + nut is called "the joint" in the following. When the joint is assembled, the arm brings it to a fixed tool. At this point, the human guides the robot motion to insert the joint into the tool. The tool allows moving the cage + nut inside the spindle and opening the empty balls sockets. The robot opens each of the empty sockets, the operator places a ball in the socket and the robot completes the ball insertion by closing the socket (Figure 8.5). When all the balls have been inserted, the robot checks that the joint rotates correctly. Finally, the operator inspects the joint one last time. If the joint is good, the operator instructs the robot to place it on a conveyor. If not, the robot places it on the scrapheap.

Though the assembly cycle is straightforward, it requires a robot with numerous information and a complex architecture. Moreover, the close collaboration between the operator and robot calls for robust security measures. The following section briefly goes over important concepts

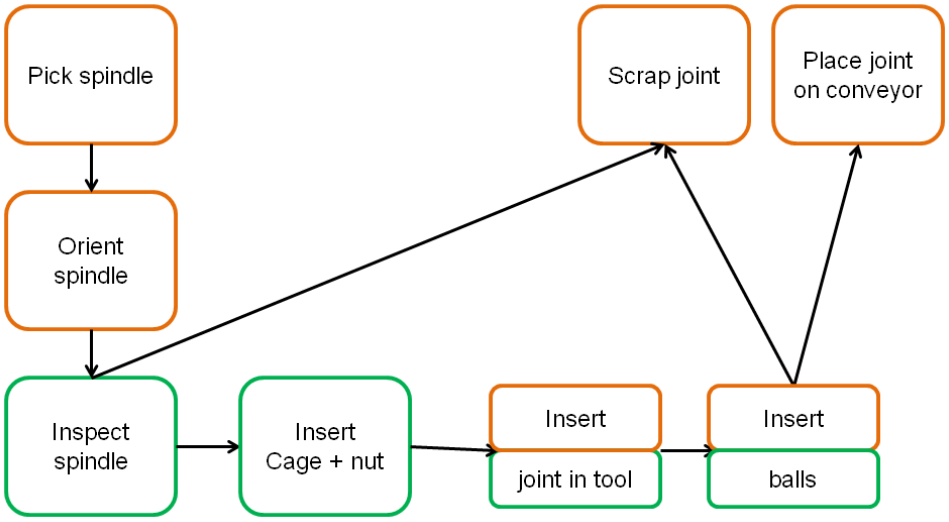
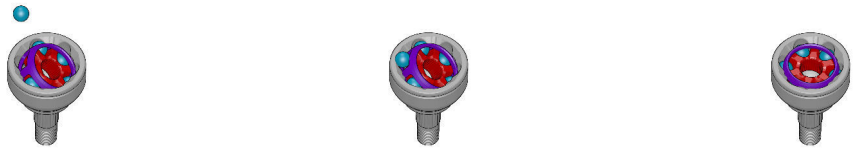


Figure 8.4: Cobotics Rzeppa joint assembly process. In orange the tasks done by the robot, in green the ones executed by the human. Two-coloured tasks are done in cooperation.



(a) Robot opens socket (b) Human places ball (c) Robot closes socket

Figure 8.5: The ball insertion process.

when talking about robots and Section 8.4 presents a generic robotic architecture illustrated with the one developed in the ICARO project.

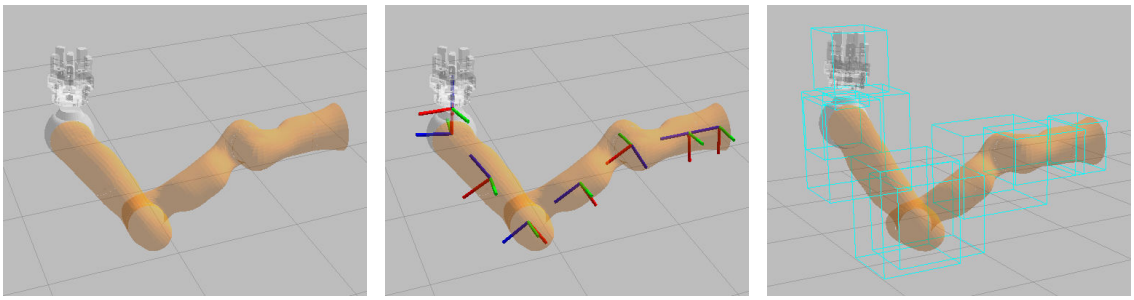
8.3 Robot and World Models

Most robots do not have the capabilities to autonomously explore themselves. So, one must provide them with various models containing different informations about their configuration. There are five crucial models: the kinematic, dynamic, inertial, collision and visual models. Apart from itself, the robot must get to know its environment. This one can be divided into a static environment and a dynamic one. Before presenting the models, Table 8.1 introduces some useful vocabulary.

The kinematic model contains the position of the joints, relative to each other, organised in a tree (Figure 8.6b). Each joint is qualified by a type, the most common types in robotics are revolute, i.e. a rotation around one axis, and translation, translation along one axis. This model bridges the position of the robot in axis space with its position in Cartesian space through inverse (Cartesian to angular) and forward (angular to Cartesian) kinematics. Kinematic data typically allows planning motions in axis or Cartesian space [18] and retrieving the arm pose space by reading the motors coders data.

Table 8.1: Some useful vocabulary.

Joint	Moving part between two segments of the robot
Link	Fixed part joining two joints, or a joints to an end effector
Axis space	Space where a point describes the angular positions of the robot joints (dimensions = number of joints).
Cartesian space	Space where a point describes a pose in Cartesian coordinates (6 dimensions).
Path	Set of ordered points in space
Trajectory	Path with evolution laws (speed, acceleration, time, etc.)
Workspace	Volume of space accessible by the robot's end effector



(a) Visual model (b) Kinematic model overlayed (c) Collision model overlayed

Figure 8.6: An illustration of the visual, kinematic and collision models.

The dynamic model is crucial for motion. It contains the nominal and maximal velocities, accelerations and jerk for each joint. It bridges the dynamics of the joints and the dynamics of the links. This is essential for control, e.g. to transform a path into a trajectory [155].

The inertial model includes the nominal and maximal torque for each joint and the inertial data for all links. This model links the efforts measured at each joint with the efforts on any part of the robot. It allows collision control and can enforce speed limits through force sensing [156]. It can also be used to replace force sensors [157].

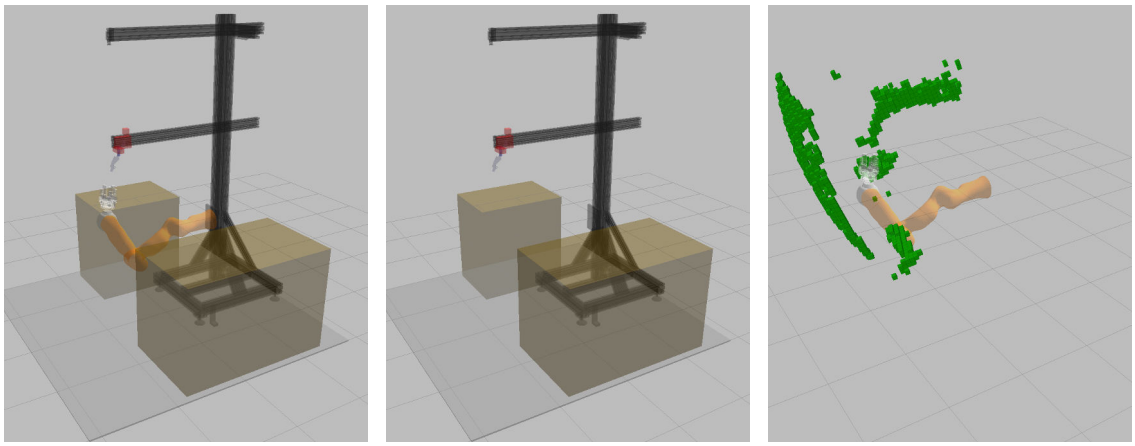
The collision model contains a simplified geometry of the robot. The 3-D model of each link can be used but usually the links are replaced by bounding boxes to simplify collision checking (Figure 8.6c). Moreover, a safety distance (padding) is often defined to keep clear from obstacles. This model is used to check for self-collisions and collisions with the environment. The collision model is a central piece of information for motion planning.

The visual model describes the appearance of the robot when visualized virtually. It is made from the textured 3-D models of the robot links (Figure 8.6a). This model is basically used for virtual visualisation or feedback. But it also allows self filtering the robot from a point cloud and it can be used for the robot to recognise itself.

The ICARO robot uses a ROS architecture. In such architecture, all the models are combined in a single URDF file. This file is loaded once when the robots starts.

8.3.1 World

The robot knows the world through a model and perceives it through its sensors. It is divided into two categories, the static part and the dynamic part.



(a) Whole world: Arm + Structure (b) Static part: Structure (c) Dynamic part: Arm + Point cloud

Figure 8.7: Different parts of the world.

The static part is composed of the fixed parts of the world, like the walls, floor and ceiling. Parts which do not move frequently, like the furniture, also belong to the static world. Usually, these are mapped once and for all when the robot is brought into its working environment (Figure 8.7b). In the present case the static world is composed of tables, the floor, aluminium frames and virtual security walls. These are described in a file containing the pose of the parts of the world relative to each other and to the robot.

The dynamic elements part is made of the moving parts of the world. Usually this corresponds to the living beings and the small objects frequently moved around (Figure 8.7c). The dynamic world is saved in a voxelmap which feeds a collision environment. This one is available for any module to query for data. The path planner is the main user of the collision environment as it checks for collisions along its paths. In this project, the dynamic elements are the human and the joint being assembled.

8.4 Robot Architecture

In this section, a generic robot architecture is introduced, Figure 8.16, and illustrated with the ICARO setup. It is divided in three parts. The low level, which includes the hardware and the hardware dependent software. The medium level, a software part implementing various algorithm hardware and task independent. The high level part, only dependent on the task at hand. In the following, for each part, the main components used in a robotic project are presented and a detailed illustration from the ICARO project is provided.

Hardware

The hardware includes motors, links and sensors. Most arms are at least equipped with position sensors.

A note about degrees of freedom: an arm with six degrees of freedom has one solution for any point in space at arm's reach. This is enough for robots who are alone in their workspace. However, in cases where there can be obstacles, static or dynamic, the configuration that allows reaching a given point may be blocked. To handle this, a seventh degree of freedom is helpful. It

provides various possible configurations for every point in the robot workspace. If one solution is blocked by an obstacle, it is possible to find a new one. The hardware used in the ICARO project is presented Figure 8.8. It can be seen that the working environment is extremely constrained thus the importance of the seventh degree of freedom.

Motor Controllers

The motor controllers constitute the interface between the digital command from the computer and the analog signal sent to the motors, though recent motors directly integrate the controllers in their hardware. The most usual inputs are a reference position, a reference velocity or impedance (position + stiffness). Usually a proportional-integral-derivative (PID) controller is involved. It allows reaching the reference quickly with a small error and few oscillations. Moreover , the motor controllers are the keeper of the motors wellbeing. They read from the dynamic model the maximum velocity and acceleration and make sure to stop the motion when they get too high.

The motor controllers can take care of other types of security checks, for example the KUKA-LWR arm is equipped with force sensors. The arm controller reads the inertial model of the robot and monitors the force sensed at each joint. By matching an estimated force with the force sensor readings it makes sure that no external force is applied to the robot. If a high external force is detected, the motor controllers stop the robot. In this project, the motor controllers have also been coupled with the output of a hand detection algorithm, see Figure 8.9. When the robot and the human are manipulating the joint together, a vision system makes sure the hands of the human are far before allowing an arm motion.

Low level abstraction layer

It is interesting to define an abstraction layer at this level to ensure hardware independence. This layer bridges the low level part, hardware dependent, with the medium level part, hardware independent. The abstraction layer itself is hardware dependent and as such belongs to the low level part.

Online trajectory generator

The online trajectory generator task is to convert a path into a trajectory. Using dynamical constraints, the trajectory generator finds the fastest trajectory along a given path. For each point of the trajectory, it provides velocity, acceleration and time.

The ICARO project relies on the Softmotion library [155, 158]. It uses cubic polynomials curves to define trajectories with bounded jerk, acceleration and velocity. As shown in Figure 8.11, Softmotion uses a seven segment acceleration pattern. Bounding the velocity results in safe motions while bounding the acceleration and jerk provides predictable and human friendly motions. It is important to note that there is always an error between the planned path and the generated trajectory. This error's bound depends on the dynamical constraints. So the path planner needs to provide a collision free tube around its path, where the tube's radius depends on the dynamical constraints (Figure 8.10.) The trajectory generator makes sure the trajectory remains in this tube.

Position Manager

The position manager stores the known position of all entities along time. The static positions are provided by the kinematic model of the robot and the model of the static world. Dynamic

positions are provided by sensors, for example position sensors at each joint provide the position of each joint and link of the robot. Visual sensors provide the pose of known objects in the world.

In the present case, a visual object detector based on the linemod algorithm, described earlier, recognises and localises the spindle to be grabbed, see Figure 8.12. Online acquisition of the spindle pose allows avoiding the calibration of the spindle conveyor position. Moreover, if for some reason a spindle arrives with a pose that prevents the robot from grasping it, the system can alert the human. At the software level, the position manager role is taken by the tf software, from the ROS library [104].

Collision Environment & Checker

The collisions environment maintains a state of world where the dynamic and static parts are represented. It is based on the collision model, which provides the static data. The dynamic part is usually provided by the position manager, by sensors or by other pieces of software. The collision checker can be queried to know if two elements from the collision environment are currently in collision or would be if placed at a given location, for example on a planned path.

In this project, there are three sources of dynamic data in the collision environment. First, the positions of the arm joints provided by the position manager. Second, an occupancy grid filled and updated by a depth sensor. For performance reasons, this occupancy grid is stored in an octomap [15]. The octomap has an updating speed which must be tuned cautiously. If the updating speed is too high, noise gets in easily, if too low, the reaction time to world's changes is slow. Third, the supervision part, when an object needs to be grasped. In this case, the collision detection is deactivated to avoid detecting a collision at grasping time. Then, the object is removed from the collision environment and added to the robot model. This allows future path planning to take into account the grasped object. This also avoids detecting permanent collisions between the object and the gripper. When releasing the object, the object is detached from the robot and returned to the collision environment. In this case, the collision checking is done by the $Kineo^{TM}CollisionDetector$ [18], a fast and exact collision checker.

Planner

The planner takes as input a start point and a goal point in space and outputs a path from start to goal. This path is required to be collision free, so the planner communicates intensively with the collision checker. A general approach for planning is to randomly sample the space with collision free paths while respecting some optimisation constraints, usually the path length.

In the ICARO project, planning is done by the $Kineo^{TM}PathPlanner$ (KPP) [18]. The particularity of this planner is its ability to perform reactive planning. This planner embeds a machine state to monitor the produced paths. If an obstacle appears across the path, the KPP monitor requests a new path from the current position to the goal which avoids the new obstacle (see Figure 8.13). Where many planners would collide, or at best get stuck in front of the obstacle, the KPP finds a new path around the obstacle and the arm continues its motion.

High Level Abstraction Layer

This layer bridges the medium level part, task independent, with the high level part, task oriented. The abstraction layer itself is task independent and as such belongs to the medium level part.

Supervision

The high level functions synchronise and monitor the high level abstraction layer tasks in order to accomplish the task at hand. When needed, the supervision asks for the human input before performing a task and returning the outputs of its action.

In this project, a machine state regroups the eight steps of the joint assembling process, cf. Figure 8.14. At almost any step, the user can ask for the joint to be scrapped. All states can lead to the errorSystem state where an error recovery strategy can be decided before continuing. From a software point of view, the machine state is set up using the Smach library, from the ROS library [104].

User Interface

The user interface is crucial in cobotics as it allows communication between the human and the robot. It includes human-to-robot communication, from keyboard to voice command, as well as robot-to-human communications, like displays, sounds or lights.

In the ICARO project case, human/robot exchanges are handled through a gesture recognition system and a screen displaying the robot world. The three available gestures *Continue*, *Stop*, *Scrap*, are shown Figure 8.15. Gesture recognition is done through two Viola-Jones detectors [131] for the *Continue* and *Scrap* gestures. The *Stop* gestures is read by tracking the user position: when the user goes out of its working zone, the robot stops. The display is made through the RViz program, from the ROS library [104], which allows visualizing the robot world and its updates (Figure 8.7).

8.5 Lessons Learned

The scientific outcomes of this project are few, and there are no quantifiable results. The valuable results are a set of lessons one should keep in mind when designing any robotic system.

Integrate in a Working Architecture

The previously described control and planning architecture is the final version reached after three working iterations. The early project architecture relied on the default ROS planner, OMPL, and the Reflexxes controller [159] (Figure 8.17a). When the Kineo planner software was ready it replaced the OMPL part. The planner, trajectory controller and interface were modified as a result (Figure 8.17b). Later on, the Softmotion software was added to the architecture and replaced Reflexxes and the trajectory controller (Figure 8.17c). The three versions were fully functional and allowed testing the contributions as they were added.

When pneumatic actuator, don't forget the tubes

When using a pneumatic actuator, unless the robot has internal tubes, one need to rely on external tubes. However, these can be burdensome as they are not accounted for in any model of the robot. For the planner they don't exist, so they can get stuck on parts of the robot during a motion. They are not considered in the collision environment, so if the depth sensor which updates it sees them, they will be considered as an obstacle. They can also hinder a vision task by occluding the target. Finally, as their state is not known, they can coil around the arm and get detached during a lengthy motion.

Dynamic is good, static too

Fast motion planning is good for reactivity and can be desirable in some situations. However, when a plan must be repeated, it is preferable to rely on precomputed paths. Moreover, this facilitate the human robot interaction as the robot is more predictable.

Invest some time in automatic calibration method

A large part of current robots are blind, they don't see the environment, its configuration must be provided manually. However, this configuration can change during the development or when the robot is moved. In this case, the environment needs to be calibrated all over again. And this happens quite often. Object detection methods are very useful in this situation to localise the different parts of the environment with respect to the robot. However, this requires modelling the environment after each modification, which can be tedious. Fast and simple modelling methods are a big plus. Investing time in developing automatic calibration methods is a time saver.

Specifically, the hand-eye calibration is crucial when manipulation is involved. It provides the robot's end effector position with respect to the sensors. This can be done automatically by having a localisation algorithm capable of localising the arm, or part of it. Calibrating the environment for collision checking purposes may not be useful. Recent sensors and methods allow scanning the environment with a depth sensor and obtaining a voxel map. However, this voxel map needs to be registered in the arm frame. Though useful, these methods have not been implemented in the present case.

Vision fails

Vision systems are very sensitive to lighting and colors. When the robot location is changed, it is necessary to check the vision first as it is highly likely that it needs tuning. For sensors like cameras, changing parameters might be sufficient to restore them. However, some sensors, e.g. active sensors, might need hardware modifications to prevent them from being blinded by a light or avoiding reflections on objects. In the ICARO setting, the arm and gripper are specular. A stocking allowed hiding these parts from the cameras. Moreover, a black sheet was laid on the floor to avoid reflections.

Plan to address complexity

When integrating functionalities to a robot, start with the essentials and make it solid enough so it will always work. Then, it is possible to build upon it by having successive functional versions of the experiment with more and more functionalities. If something does not work, just fold back to the previous version.

For example, in this project, the first version just grasped the spindle and moved it from task to task with the human actually performing the tasks. A more developed version saw the arm perform the spindle orientation and releasing it on the exit or scrap conveyor. However, the balls insertion was entirely done by the human. Then, an advanced version took also care of inserting two balls. The four remaining balls being inserted by the human. The finally version adds the insertion of the six balls.

Plan for accidents

When a potential danger is identified, three securities need to be set up. The first one to anticipate the danger, this security should avoid any accident. The second to reduce the consequences

of an accident, in case something happens, the damages should be as small as possible. Third, to recover after the accident, in order to allow the damages to be repaired as soon as possible. This philosophy is illustrated in the ICARO project by the security around the ball insertion process. A camera prevents the operator from catching its fingers when closing the sockets. However, if this happens a high external force will be detected by the arm and it will stop moving before crushing the fingers. Finally, the arm enters impedance control mode which allows gently pushing it back so the human can pull out its fingers. A full security analysis has been done by the PSA teams which considers every potential hazard.

This chapter introduced the problems involved in assembling an Rzeppa joint and proposed a cobotic solution in the context of the ICARO project. After presenting the problem, it proposes a roadmap pointing to the necessary elements of a robot, with illustrations, and to the cautions one should take when building such robot. Though little research is involved in this project, it is a perfect example of how readily available techniques can be combined to build a robot for an industrial application. Due to its satisfactory degree of completion, this project has been presented in various exhibitions including the SIANE robotics fair and the French Ministère de l'Economie Cobotic Contest 2015.

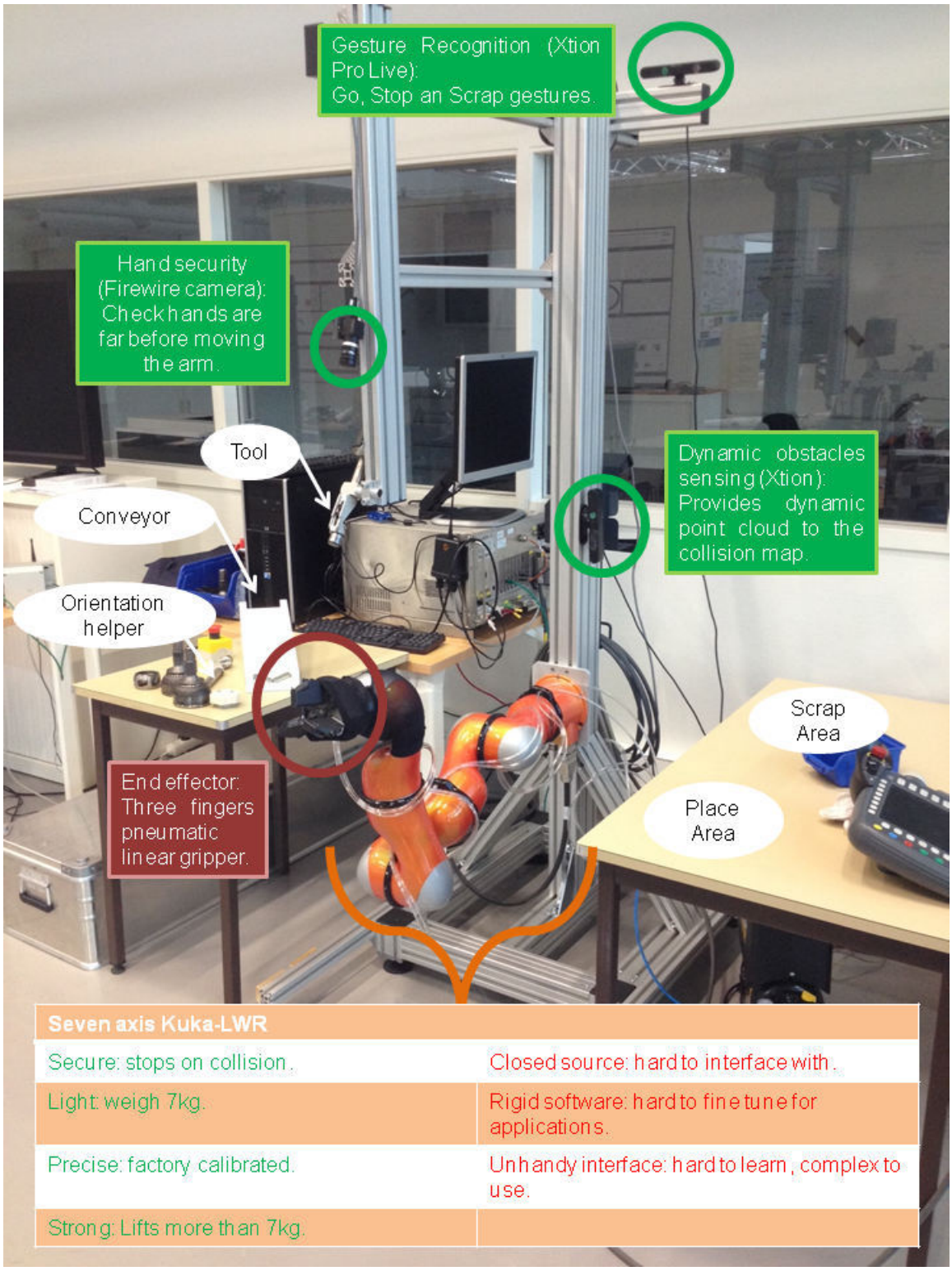


Figure 8.8: A robotic software architecture.

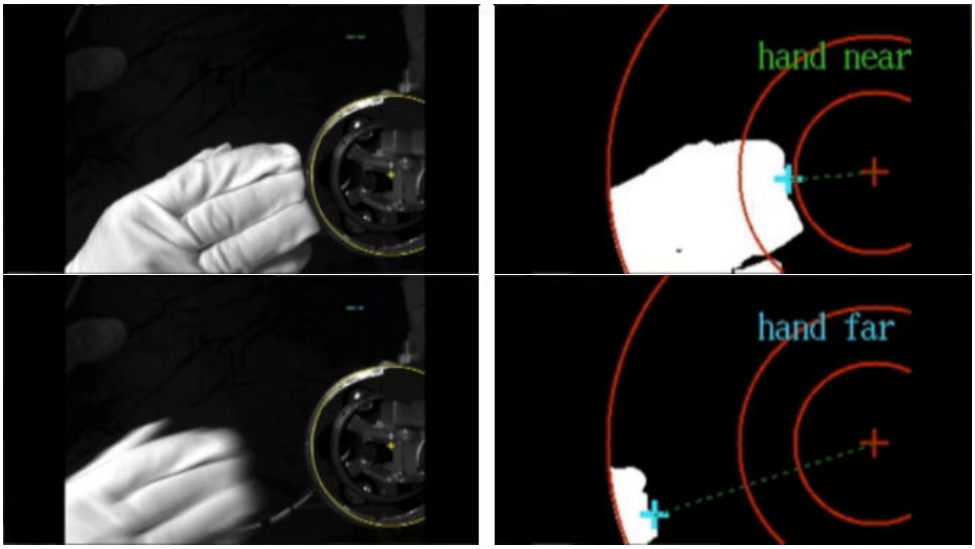


Figure 8.9: Hands monitoring for security.

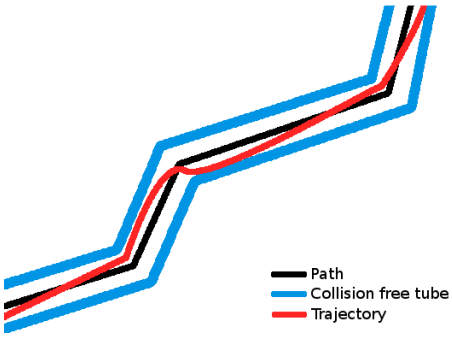
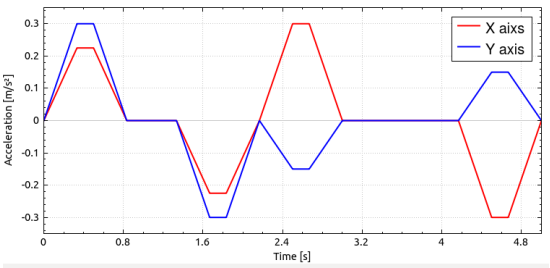
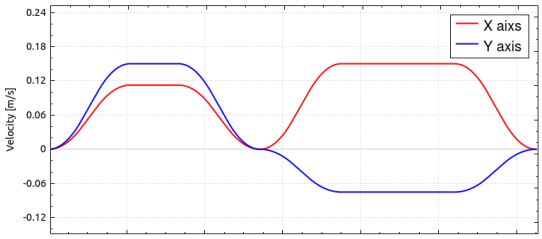


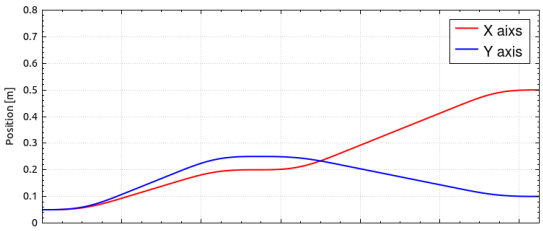
Figure 8.10: Illustration of a trajectory differing from the initial path but remaining in a bounded tube.



(a) Acceleration



(b) Velocity



(c) Position

Figure 8.11: Seven segments control pattern: increase acceleration, maximum acceleration, reduce acceleration, null acceleration, deceleration, maximum deceleration, reduce deceleration to zero. Resulting velocity and position are also provided.

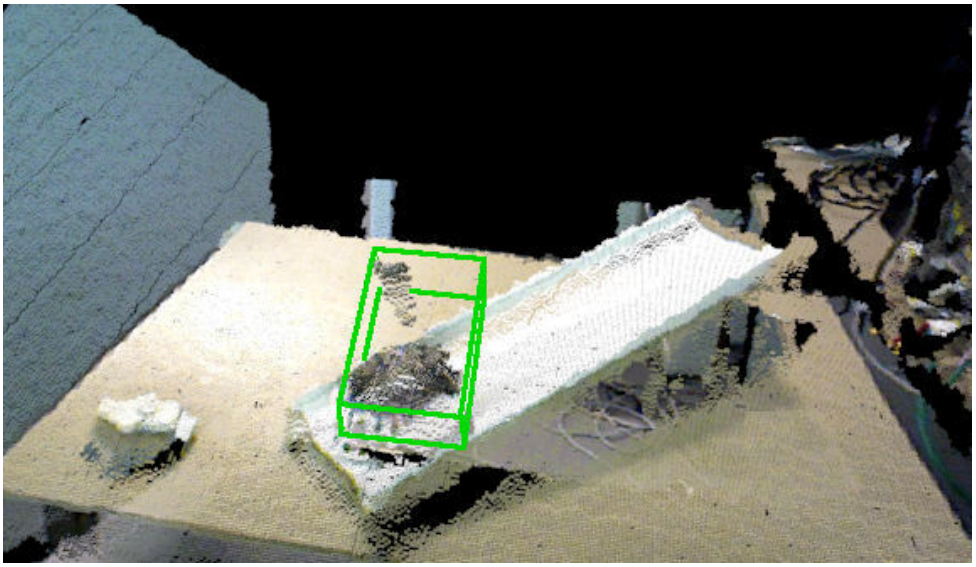


Figure 8.12: The linemod algorithm computes a bounding box (green) around the visible part of the spindle. This gives the spindle location, but also notifies if a spindle is present on the conveyor.

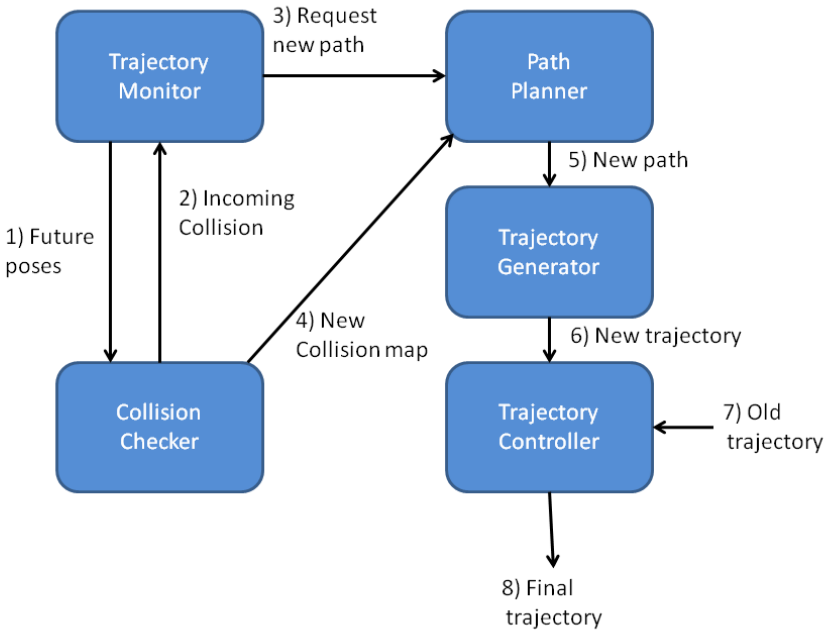


Figure 8.13: The trajectory monitor supervises the current trajectory and checks if future position are in a collision state. In case of incoming collision, it requests a new path to the planner. The new path is directly sent through the trajectory generator and to the controller. The controller merges the old trajectory with the new one to obtain a smooth transition while dodging the obstacle.

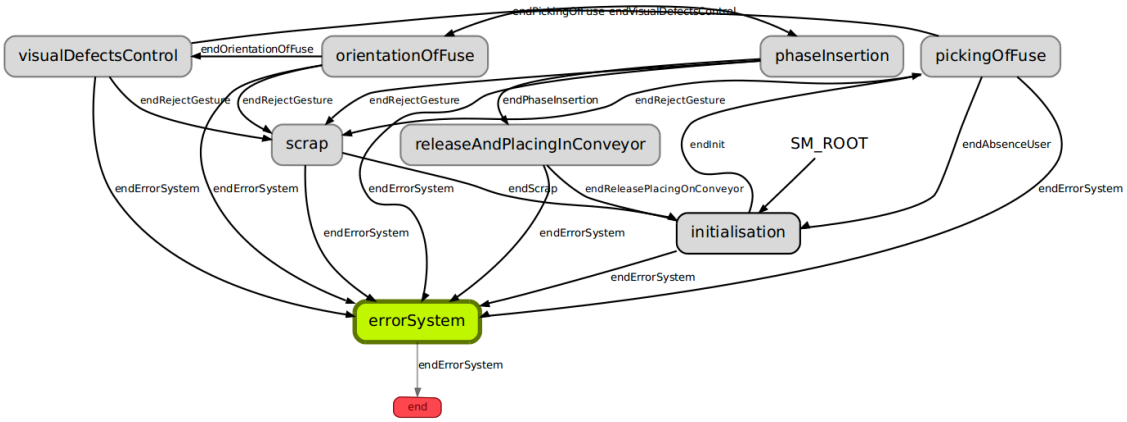


Figure 8.14: The machine state has eight states: initialisation, picking of fuse, orientation of fuse, visual defects control, phase insertion, release and placing in conveyor, scrap and error system.

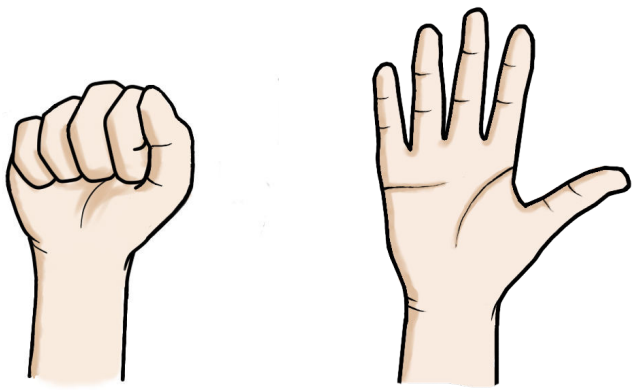


Figure 8.15: *Start* and *Scrap* gestures.

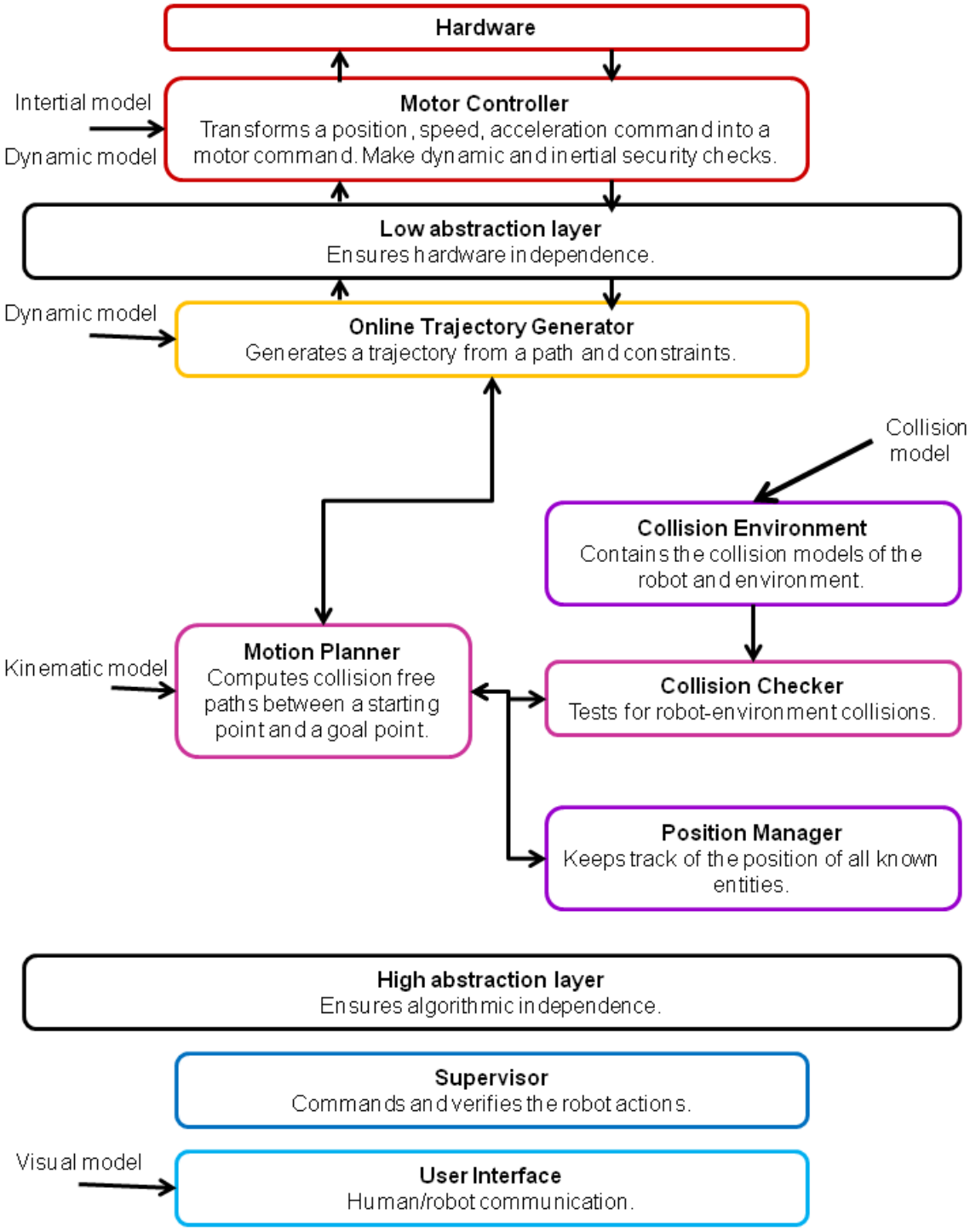
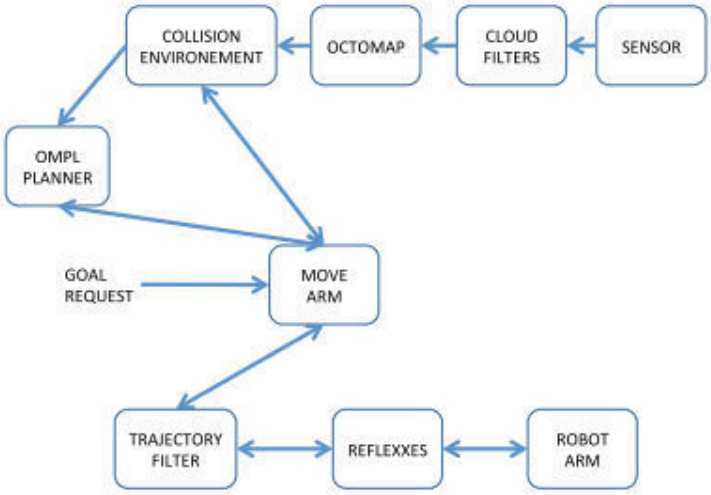
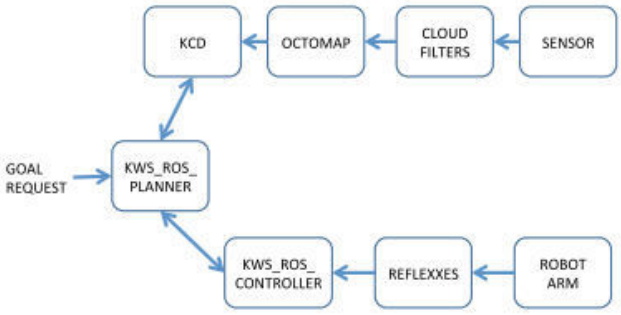


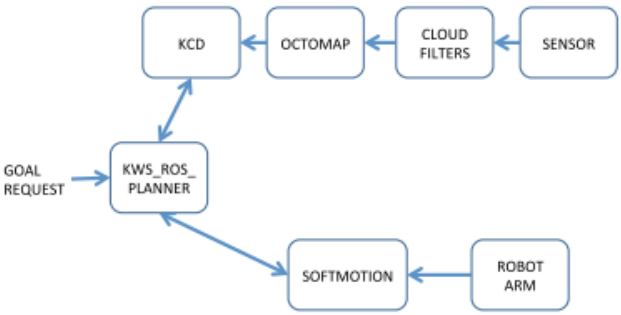
Figure 8.16: A robotic software architecture.



(a) OMPL and Reflexxes



(b) Kineo Works and Reflexxes



(c) Kineo Works and Softmotion

Figure 8.17: The planning and control architecture evolved through three steps as the partners contributions were integrated.

Conclusion

Pour atteindre l'autonomie, les robots doivent percevoir le monde. Ce monde se separt en une partie statique et une partie dynamique. Bien que la partie statique puisse être apprise une fois pour toute, la partie dynamique doit être mise à jour en permanence. Bien que des algorithmes existent pour cela, leur utilisation dans le monde de la robotique est encore marginale. Nous pensons que cela est du à la modélisation des objets qui est trop difficile.

Dans le Chapitre 1 de cette Thèse, nous présentons le principe de processus de modélisation qui permet de créer une pipeline de modélisation. Cela nous mène à choisir la modélisation à base de descripteurs locaux texturés pour le reste du travaille.

Puis le Chapitre 2 présente quelque méthodes de localisation et reconnaissance ainsi qu'un état de l'art de techniques de modélisation. Nous montrons que ces techniques ont deux défauts, elles sont trop difficiles à utiliser et produisent des modèles trop complexes.

Le Chapitre 3 montre qu'il est possible de produire de modèles plus simples en utilisant une quantité limitée d'information et les Chapitre 4 et 5 donnent des moyens simples de créer ces modèles, répondant ainsi au deux problèmes soulevés précédemment.

Dans les Chapitres 6 et 7, nous soulevons l'utilité de l'utilisation du contexte pour faciliter la reconnaissance d'objets et de catégories d'objets. Le Chapitre 6 se concentre sur l'utilisation du lieux et montre comment un robot peut explorer un lieux puis associer des lieux privilégiés à certains objets. La suite montre comment utiliser cette information pour faciliter la reconnaissance en mixant contexte et descripteurs simples dans une cascade. Le Chapitre 7 quand à lui montre une manière d'apprendre le contexte objet-objets depuis Amazon.com et de l'utiliser avec des méthodes graphiques de manière simple en utilisant des Reseau de Markov Logiques. Une experience est proposée mais pas réalisée.

Pour finir les Chapitre 8 montre une application robotique complète dans laquelle un système de reconnaissance et localisation d'objets est intégré. Nous présentons l'intégralités du système et comment les différentes parties intéragissent.

Ce travaille peut être poursuivi dans de nombreuses direction. Les modèles simples proposés ont des angles morts et nécessitent une camera RGB-D. En bougeant le robot, les angles morts peuvent être ignorés. Par ailleurs, en utilisant le tenseur trifocal la contrainte RGB-D pourrait être levée. Dans le cas de l'utilisation d'Amazon.com pour le contexte, il serait souhaitable d'utiliser cette information comme une base sur laquelle construire avec des méthodes d'apprentissage en ligne. Pour finir, nous pensons que pour les modeles d'objets ou le contexte, un robot devrait être livré avec une base de données initiales, contenant principalement des categories d'objets et des informations générales, et que cette base doit être enrichie en ligne lors de la vie du robot à mesure qu'il rencontre des instances et qu'il découvre le contexte qui l'entoure.

Conclusion

Perceiving objects is crucial for robots to attain autonomy. Multiple steps are required for a robot to sense objects and complex pipelines yielding good results have been proposed in the state of the art [54, 35, 5]. Though, many robots, in research communities and industrial applications, remain blind. We believe that the blocking point is the modelling process: the learning stage through which the system gets to know new objects and their properties.

In Chapter 1, the methodology to build a perceptual model has been separated into four parts: world constraints, sensors, object properties and descriptors; our choice for each of these elements has been explained as follows. The properties of an object useful for a robot are the objects' identity, its pose in space and possibly its structure. These can be measured with RGB and RGB-D cameras, so the focus has been put on visual perception. The main constraints for robotics manipulation have been considered to be pose change, occlusion, lighting, multiplicity, speed, and precision. To answer these, the existing visual descriptors have been compared against these constraints and it appeared that the local texture descriptors provide the best compromise.

With a framework for the modelling process, Chapter 2 took a look at standard modelling pipelines to find out what are the limitations that make these approaches unpractical. First, the multiple views paradigm has been presented, where a model is built out of numerous views, along with the ways to acquire these views, online or virtually. While online methods provide a more realistic model, for practical reasons virtual approaches have been privileged. To create a model from virtual views, a textured CAD model of the object should be available. The question of how to build this CAD model has been answered with a review of the Structure from Motion, Simultaneous Localisation and Mapping and Depth Sensing methods. Two readily available methods have been compared, a Structure from Motion one and a Depth Sensing one, each with its strength and weaknesses. Selecting the correct method depends on the descriptors at hand. To build the model from a virtual mesh, various sampling approaches have been presented and the choice of the golden spiral has been justified by the higher density of the observation sphere coverage. This Chapter concluded on the fact that the resulting models tend to be complex. Moreover, it showed that the state-of-the-art methods are difficult to use and require expertise.

The models' complexity can be mitigated by balancing the number of views with the descriptors' robustness, as shown in Chapter 3. The robust descriptors paradigm is introduced and an experiment to compare existing textured descriptors is presented. The results showed that the ASIFT descriptor provides the higher robustness. A second experiment assessed how many views are needed to model an object in a robust way with ASIFT. The viewpoints from which an object is not detected with a given model are called the model's blind spots. The results suggested that as few as seventeen views provide a model with small blind spots, a reasonable size and small processing time. This Chapter finished by proposing the concept of blind spot aware models. These models can provide to planning algorithms the blind spots of the model, so

they can be taken into account when planning motions or tasks that require to keep an object localised.

The remaining problem, object modelling difficulty, has been tackled in Chapters 4 and 5. First has been proposed a method to model objects from a limited set of images, the object dimensions and its shape. Though some approximations are made in the modelling, the resulting localisation seemed precise enough for grasping. To push further the idea of modelling from images, we proposed to simply use descriptors from images as model, and no other information. This required to design a new localisation approach to combine the input from a RGB-D sensor with these descriptors. The localisation method has been evaluated in simulation and it exhibited comparable results with state-of-the-art PnP algorithms, in the calibrated and uncalibrated cases. Finally, results for the calibrated case with real objects have been provided as illustration.

The previous Chapters treated of learning perceptual models to localise objects. The next problem has been how to recognise them. In service robotics, recognition needs to handle tens of thousands of objects; in the industrial case, the number of objects is low but each object can appear under numerous variants (size, assembly completeness, etc.). In both cases the process should have high speed and precision. In order to provide such performances, visual information may not be sufficient. We believe a robot can retrieve helpful information from its context. Two types of contexts have been considered, the place and the co-occurrent objects. For each, the learning and inference problems are addressed.

To autonomously learn relationships between places (e.g. rooms), areas (e.g. parts of furniture) and objects, Chapter 6 proposed an autonomous exploration, modelling, segmentation and classification strategy. This method has been illustrated in the case of the ADREAM apartment. The method was able to segment the different rooms and find if they were linked through doors or windows. The planar areas from each place have been segmented as objects are likely to sit there. This provided the places-areas relationships. Moreover, the robot went around the apartment and learned the probabilities to find known objects in the previously segmented areas. This yielded the objects-areas relationships. Results obtained in a real situation with readily available software provided interesting results with nearly half of the objects-areas links correctly found. Having learned the likelihood to find objects in given places, the second part of this Chapter focused on merging this data with visual information. A cascade has been proposed to merge weak visual descriptors and contextual data. Results showed that this approach allowed discarding from 50% to 98% of unlikely candidates when performing recognition.

In order to improve on the methods presented above, Chapter 7 tackled the learning and inference of co-occurent objects. The first part presented a method to automatically learn object-objects co-occurrence matrices for lists of objects. By querying Amazon.com for selling informations, it was possible to build co-occurrence matrices for objects. Three different types of matrices could be obtained, binary, based on apparition frequency, based on selling ranks. These matrices could then be merged with visual information with a Markov Network. In order to facilitate the design of the Markov Network with complex relationships, we proposed to use a Markov Logic Network. This approach allows building a Markov Network from a set of first order logic formulas. A concrete case has been presented using the COCO dataset, with the electronic and kitchen super-categories. The corresponding co-occurrence matrices from Amazon.com have been provided and a set of logic formulas allowed designing a Markov Logic Network. No results could be provided for this chapter as the COCO datasets is not ready yet.

Finally, Chapter 8 showed a complete robotic system for collaborative joint assembling. This allowed putting in contrast the work of this Thesis in a practical situation. A general approach

to robots architecture have been presented and the particularities of this system have been used as illustration. The outcome of this project being difficult to assess, a set of lessons learned have been provided as a result.

To summarise, we have shown that perfect models are hard to create, complex in terms of size, loading speed and processing speed. We believe one should favour simple models aware of their own shortcomings. We showed that modelling methods relying on small sets of images are easier to automate than scanning based methods. Two such methods were proposed to model from images with or without a priori information.

For context modelling, we introduced two learning methods allowing respectively to learn place-objects relationships from experience and object-objects relationships from the Internet. Corresponding inference methods were proposed based on cascades of weak descriptors and graphical models. It has been shown that such inference using the context and simple descriptors facilitate the recognition step.

Overall, the stress is on automatic methods. The structureless model allows automated modelling from the internet, the environment exploration method allows autonomously learning objects-places context, the CBIB feature allows learning object-objects context from Amazon.com and the Markov Logic Networks allows building graphical models automatically from logic formulae. The structureless models and learning from Amazon.com are of particular interest as they are linked to the big data problematic where one try to extract useful information from bulk of data available on the web.

In this Thesis, we have proposed to use simple perceptual models made of a limited set of images with no overlap or position a priori. Though this yields light models, this approach suppose that the sensor can move. So a robotic system is needed to handle the case where the sensor is in the object's blind spot. In any case, we believe a robot should not be static when observing a scene. It should move its sensors or its whole body to observe the scene from different viewpoints. To this end, robots should be equipped with easily moving sensors, like the cameras in PR2's wrists, and take them into consideration when planning the exploration of a scene. This way, while exploring a scene, the robot can recognise and localise even light models.

These light models contain a small number of images from which local textured features are extracted for recognition and localisation. The precision and discriminative power of such approaches depend on the image resolution. The higher the resolution the more precise the estimations. Though, our method needs an input from a depth sensor and most popular depth sensors (Kinect, Xtion, etc.) provide a low resolution of 640x480.

A direct solution to this problem would be to rely on a more expensive depth sensor with higher resolution. A smarter solution would be to analyse the problem with the trifocal tensor in mind. Indeed, to solve this problem we used three views combined two by two with two views geometry. Using the trifocal tensor would consider the geometrical relationships between the three images as a whole. Intuition tells us that it may discard the need of a depth sensor and that the method could work with 2-D images. Such method could be compared with approaches based on the fundamental matrix to determine which yields the highest precision.

To improve on these methods, we have shown that context can be crucial. However, having a user teaching contextual information to a robot is a challenging tasks as robots capabilities in terms of human perception and understanding are still limited. Learning it from Amazon.com is convenient, but only provides partial information. Indeed, the CBIB feature results are computed

with unknown rules that may make sense for a recommender system but not necessarily for contextual information. Moreover, it represents the habits of a fraction of the population, which may not be representative of the robot's world.

Most learning methods for context are supervised, they use labelled examples. Though learning from unlabelled data is possible, it is much more effective if some initial information is available. However, if the initial information is false, this can break down the whole learning process. For these reasons, we believe robots need to be shipped with a very general database of contexts. This one should be updated and adapted to the robot situation with a web based learning method, to make sure the initial data is correct. Finally, autonomous learning throughout the robot life will enrich the context database to adapt to the more common situations encountered by the robot.

This work approached the places and objects as a source of context. Another rich source of context is the human. Perceiving human actions and the use they make of objects can greatly improve recognition performances. In any case, new ways to learn contextual information, visual or other, from numerical data are still needed.

We think that perceptual and contextual models need to be learned continuously throughout the robot's life. It should be done autonomously as few users would accept the burden of spending hours teaching new models to the robot. In this perspective, we believe this work provided interesting insights and realistic solutions for learning from the largest data source available to both humans and machines : the Internet. Indeed, the proposed methods successfully move the onus of modelling from the user to the robot.

From a general perspective, object modelling, recognition and localisation are building blocks necessary to focus on higher level cognitive task. For example, semantic SLAM, where a robot localise itself by using known objects as landmarks; learning by example, where a robot learns a task by observing a human doing it; or the reverse problem, where the robot teaches a task to a human and monitors him to make sure it is done properly. These two last research axis are of particular interest as they both find numerous applications. They form the basis of the future companion robot able to learn new tasks from the human and to teach him new knowledge.

Publications List

MANFREDI, Guido, DEVY, Michel, et SIDOBRE, Daniel. Accélérer et simplifier la reconnaissance d'objets avec des descripteurs visuels et contextuels simples. In : Orasis 2013, Congrès des jeunes chercheurs en vision par ordinateur.

MANFREDI, Guido, DEVY, Michel, and SIDOBRE, Daniel. Multi class object recognition with an adaptive confidence: Cascade of weak descriptors for fast hypothesis elimination. In : Electronics, Control, Measurement, Signals and their application to Mechatronics (ECMSM), 2013 IEEE 11th International Workshop of. IEEE, 2013. p. 1-4.

DUMONTEIL, Gauthier, MANFREDI, Guido, DEVY, Michel, CONFETTI, Ambroise, and SIDOBRE, Daniel. Reactive Planning on a Collaborative Robot for Industrial Applications. In : 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2015) p.(to appear).

MANFREDI, Guido, DEVY, Michel, and SIDOBRE, Daniel. Textured Object Recognition: Balancing Model Robustness and Complexity. In : Computer Analysis of Images and Patterns. Springer International Publishing, 2015. p. 52-63.

MANFREDI, Guido, DEVY, Michel, and SIDOBRE, Daniel. Visual Localisation from Structureless Rigid Models. In : Advanced Concepts for Intelligent Vision Systems. Springer International Publishing, 2015. p. 510-520.

Bibliography

[1] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M M Montiel, A Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "Roboearth," *Robotics Automation Magazine, IEEE*, vol. 18, no. 2, pp. 69–82, June 2011.

[2] John Oliensis, "A critique of structure-from-motion algorithms," *Computer Vision and Image Understanding*, vol. 80, no. 2, pp. 172–214, 2000.

[3] Noah Snavely, Steven M Seitz, and Richard Szeliski, "Photo tourism: exploring photo collections in 3d," in *ACM transactions on graphics (TOG)*. ACM, 2006, vol. 25, pp. 835–846.

[4] Krystian Mikolajczyk and Cordelia Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[5] A. Aldoma, Z. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 80 –91, sept. 2012.

[6] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin, "Registration of 3d point clouds and meshes: a survey from rigid to nonrigid," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 7, pp. 1199–1217, 2013.

[7] David Crandall, Andrew Owens, Noah Snavely, and Dan Huttenlocher, "Discrete-continuous optimization for large-scale structure from motion," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3001–3008.

[8] Sudipta N Sinha, Drew Steedly, and Richard Szeliski, "A multi-stage linear approach to structure from motion," in *Trends and Topics in Computer Vision*, pp. 267–281. Springer, 2012.

[9] Santosh Kumar Divvala, Derek Hoiem, James H Hays, Alexei A Efros, and Martial Hebert, "An empirical study of context in object detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1271–1278.

[10] Carolina Galleguillos and Serge Belongie, "Context based object categorization: A critical survey," *Computer Vision and Image Understanding*, vol. 114, no. 6, pp. 712–722, 2010.

[11] Li-Jia Li and Li Fei-Fei, "What, where and who? classifying events by scene and object recognition," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* IEEE, 2007, pp. 1–8.

[12] Myung Jin Choi, Antonio Torralba, and Alan S Willsky, "Context models and out-of-context objects," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 853–862, 2012.

[13] Bangpeng Yao and Li Fei-Fei, "Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 9, pp. 1691–1703, 2012.

[14] Kai Huebner, Steffen Ruthotto, and Danica Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on.* IEEE, 2008, pp. 1628–1633.

[15] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, Software available at `http://octomap.github.com`.

[16] Yiming Ye and John K Tsotsos, "Sensor planning for 3d object search," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 145–168, 1999.

[17] Felipe Trujillo-Romero, Victor Ayala-Ramírez, Antonio Marín-Hernández, and Michel Devy, "Active object recognition using mutual information," in *MICAI 2004: Advances in Artificial Intelligence*, pp. 672–678. Springer, 2004.

[18] J-P Laumond, "Kineo cam: a success story of motion planning algorithms," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 90–93, 2006.

[19] Jean-Philippe Saut and Daniel Sidobre, "Efficient models for grasp planning with a multi-fingered hand," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 347–357, 2012.

[20] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva, "A survey on pixel-based skin color detection techniques," in *Proc. Graphicon.* Moscow, Russia, 2003, vol. 3, pp. 85–92.

[21] David G Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on.* Ieee, 1999, vol. 2, pp. 1150–1157.

[22] Jie Tang, Stephen Miller, Arjun Singh, and Pieter Abbeel, "A textured object recognition pipeline for color and depth image data," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on.* IEEE, 2012, pp. 3467–3474.

[23] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen, "Face description with local binary patterns: Application to face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 12, pp. 2037–2041, 2006.

[24] Engin Tola, Vincent Lepetit, and Pascal Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 5, pp. 815–830, 2010.

[25] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010*, pp. 778–792. Springer, 2010.

[26] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[27] Ashutosh Saxena, Min Sun, and Andrew Y Ng, "Make3d: Learning 3d scene structure from a single still image," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 824–840, 2009.

[28] Dahua Lin, Sanja Fidler, and Raquel Urtasun, "Holistic scene understanding for 3d object detection with rgbd cameras," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1417–1424.

[29] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid, "Aggregating local image descriptors into compact codes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 9, pp. 1704–1716, 2012.

[30] Andrew E. Johnson and Martial Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 433–449, 1999.

[31] Federico Tombari, Samuele Salti, and Luigi Stefano, "Unique signatures of histograms for local surface description," in *Computer Vision – ECCV 2010*, Kostas Daniilidis, Petros Maragos, and Nikos Paragios, Eds., vol. 6313 of *Lecture Notes in Computer Science*, pp. 356–369. Springer Berlin Heidelberg, 2010.

[32] 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*. IEEE, 2009, pp. 3212–3217.

[33] Yan Ke, Rahul Sukthankar, and Martial Hebert, "Efficient visual event detection using volumetric features," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. IEEE, 2005, vol. 1, pp. 166–173.

[34] Alaa E Abdel-Hakim and Aly A Farag, "Csift: A sift descriptor with color invariant characteristics," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. IEEE, 2006, vol. 2, pp. 1978–1983.

[35] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 858–865.

[36] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. IEEE, 2006, vol. 2, pp. 2169–2178.

[37] Jean-Michel Morel and Guoshen Yu, "Asift: A new framework for fully affine invariant image comparison," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 438–469, 2009.

[38] Koen EA Van De Sande, Theo Gevers, and Cees GM Snoek, "Evaluating color descriptors for object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1582–1596, 2010.

[39] Dengsheng Zhang and Guojun Lu, "Review of shape representation and description techniques," *Pattern recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[40] Nianjuan Jiang, Ping Tan, and Loong-Fah Cheong, "Seeing double without confusion: Structure-from-motion in highly ambiguous scenes," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1458–1465.

[41] Michele Fenzi, Ralf Dragon, Laura Leal-Taixé, Bodo Rosenhahn, and Jörn Ostermann, *3D Object Recognition and Pose Estimation for Multiple Objects using Multi-Prioritized RANSAC and Model Updating*, Springer, 2012.

[42] "An initial assessment of the environmental impact of grocery products," `http://www.wrap.org.uk/sites/files/wrap/An%20initial%20assessment%20of%20the%20environmental%20impact%20of%20grocery%20products%20final_0.pdf`.

[43] Andrew J Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1403–1410.

[44] Nasser H Dardas and Nicolas D Georganas, "Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques," *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, no. 11, pp. 3592–3607, 2011.

[45] Davide Scaramuzza and Friedrich Fraundorfer, "Visual odometry [tutorial]," *Robotics & Automation Magazine, IEEE*, vol. 18, no. 4, pp. 80–92, 2011.

[46] Andrew I Comport, Eric Marchand, Muriel Pressigout, and Francois Chaumette, "Real-time markerless tracking for augmented reality: the virtual visual servoing framework," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 4, pp. 615–628, 2006.

[47] Bernard Espiau, François Chaumette, and Patrick Rives, "A new approach to visual servoing in robotics," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 3, pp. 313–326, 1992.

[48] Éric Marchand, Fabien Spindler, and François Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *Robotics & Automation Magazine, IEEE*, vol. 12, no. 4, pp. 40–52, 2005.

[49] IRISA Lagadic, "Visp," `http://www.irisa.fr/lagadic/visp/visp.html`, 2015.

[50] Hervé Jégou, Matthijs Douze, and Cordelia Schmid, "Improving bag-of-features for large scale image search," *International Journal of Computer Vision*, vol. 87, no. 3, pp. 316–336, 2010.

[51] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[52] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[54] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, p. 0278364911401765, 2011.

[55] OpenCV Fundation, "Opencv," `http://opencv.org/`, 2015.

[56] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," `http://www.vlfeat.org/`, 2008.

[57] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[58] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[59] Shiqi Li, Chi Xu, and Ming Xie, "A robust o (n) solution to the perspective-n-point problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1444–1450, 2012.

[60] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Astrom, and Masatoshi Okutomi, "Revisiting the pnp problem: A fast, general and optimal solution," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2344–2351.

[61] Willow Garage, "Tabletop object recognition," `http://wg-perception.github.io/tabletop/`, 2011.

[62] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 2155 –2162.

[63] Willow Garage, "Object recognition kitchen," `http://wg-perception.github.io/object_recognition_core/`, 2013.

[64] "Archive3d," `http://archive3d.net/`, 2015.

[65] "Grabcad," `https://grabcad.com/`, 2015.

[66] Google, "3d warhouse," `https://3dwarehouse.sketchup.com/`, 2015.

[67] ICRA, "Solutions in perception instance recognition challenge," `http://opencv.willowgarage.com/wiki/SolutionsInPerceptionChallenge`, 2011.

[68] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox, "Rgb-d object recognition: Features, algorithms, and a large scale benchmark," in *Consumer Depth Cameras for Computer Vision*, pp. 167–192. Springer, 2013.

[69] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel, "Bigbird: A large-scale 3d database of object instances," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 509–516.

[70] Michael Firman, "More rgb-d datasets," `http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/`, 2015.

[71] Peter Sturm, "A historical survey of geometric computer vision," in *Computer Analysis of Images and Patterns*. Springer, 2011, pp. 1–8.

[72] Marc Pollefeys, Reinhard Koch, Maarten Vergauwen, and Luc Van Gool, "Flexible acquisition of 3d structure from motion," in *Proc. IEEE workshop on Image and Multidimensional Digital Signal Processing*. Citeseer, 1998.

[73] Oren Boiman, Eli Shechtman, and Michal Irani, "In defense of nearest-neighbor based image classification," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[74] David Nistér, "An efficient solution to the five-point relative pose problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 756–770, 2004.

[75] Quan-Tuan Luong and Olivier D Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *International Journal of Computer Vision*, vol. 17, no. 1, pp. 43–75, 1996.

[76] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[77] Richard I Hartley and Peter Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[78] Autodesk, "123d catch," `http://www.123dapp.com/catch`, 2014, Accessed: 2010-09-30.

[79] Qi Pan, Gerhard Reitmayr, and Tom Drummond, "Proforma: Probabilistic feature-based on-line rapid model acquisition.," in *BMVC*, 2009, pp. 1–11.

[80] ReconstructMe Team, "Reconstructme," `http://reconstructme.net/`, 2015.

[81] Noah Snavely, Steven M Seitz, and Richard Szeliski, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, 2008.

[82] Yasutaka Furukawa and Jean Ponce, "Accurate, dense, and robust multiview stereopsis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 8, pp. 1362–1376, 2010.

[83] Eric Royer, Maxime Lhuillier, Michel Dhome, and Thierry Chateau, "Localization in urban environments: monocular vision compared to a differential gps sensor," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 2, pp. 114–121.

[84] Wikipedia, "Kinect," http://fr.wikipedia.org/wiki/Kinect, 2008.

[85] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[86] Occipital, "Skanect," http://skanect.occipital.com/, 2015.

[87] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox, "Manipulator and object tracking for in-hand 3d object modeling," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, 2011.

[88] Autodesk, "123d catch," http://www.123dapp.com/howto/catch, 2014, Accessed: 2010-09-30.

[89] Meshlab, "Meshlab," http://meshlab.sourceforge.net/, 2015.

[90] BlenderFoundation, "Blender," http://www.blender.org/, 2015.

[91] Dave Rusin, "Disco ball," http://www.math.niu.edu/~rusin/known-math/95/equispace.elect, 2015.

[92] Edward B Saff and A BJ Kuijlaars, "Distributing many points on a sphere," *The mathematical intelligencer*, vol. 19, no. 1, pp. 5–11, 1997.

[93] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[94] Chavdar Papazov and Darius Burschka, "An efficient ransac for 3d object recognition in noisy and occluded scenes," in *Computer Vision–ACCV 2010*, pp. 135–148. Springer, 2011.

[95] Chris Harris and Mike Stephens, "A combined corner and edge detector.," in *Alvey vision conference*. Manchester, UK, 1988, vol. 15, p. 50.

[96] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2006.

[97] Ramin Zabih and John Woodfill, "A non-parametric approach to visual correspondence," in *IEEE transactions on pattern analysis and machine intelligence*. Citeseer, 1996.

[98] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst, "Freak: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. Ieee, 2012, pp. 510–517.

[99] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua, "Fast keypoint recognition using random ferns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 3, pp. 448–461, 2010.

[100] Vincent Lepetit, Pascal Lagger, and Pascal Fua, "Randomized trees for real-time keypoint recognition," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 2, pp. 775–781.

[101] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1817–1824.

[102] Edward Rosten and Tom Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, pp. 430–443. Springer, 2006.

[103] Changchang Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[104] Willow Garage, "Robot operating system (ros)," `www.ros.org`, 2010.

[105] Tayeb Basta, I Rudas, and N Mastorakis, "Mathematical flaws in the essential matrix theory," in *WSEAS International Conference. Proceedings. Recent Advances in Computer Engineering*. WSEAS, 2009.

[106] C-P Lu, Gregory D Hager, and Eric Mjolsness, "Fast and globally convergent pose estimation from video images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 6, pp. 610–622, 2000.

[107] Joel A Hesch and Stergios I Roumeliotis, "A direct least-squares (dls) method for pnp," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 383–390.

[108] Olivier D Faugeras, Q-T Luong, and Stephen J Maybank, "Camera self-calibration: Theory and experiments," in *Computer Vision—ECCV'92*. Springer, 1992, pp. 321–334.

[109] Changchang Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[110] ASUS, "Asus xtion pro live rgbd sensor," `http://www.asus.com/Multimedia/Xtion_PRO_LIVE/`.

[111] Roger Y Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 4, pp. 323–344, 1987.

[112] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 3485–3492.

[113] Abdennour Aouina, Michel Devy, and A Marin Hernandez, "3d modeling with a moving tilting laser sensor for indoor environments," in *World Congress*, 2014, vol. 19, pp. 7604–7609.

[114] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Andreas Holzbach, and Michael Beetz, "Model-based and learned semantic object labeling in 3d point cloud maps of kitchen environments," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 3601–3608.

[115] Andreas Nüchter and Joachim Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915–926, 2008.

[116] Francesco Amigoni and Vincenzo Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robot. Auton. Syst.*, vol. 58, no. 5, pp. 684–699, May 2010.

[117] Paul S. Blaer and Peter K. Allen, "View planning and automated data acquisition for three-dimensional modeling of complex sites," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 865–891, 2009.

[118] C.J. Taylor and D. Kriegman, "Exploration strategies for mobile robots," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, 1993, pp. 248–253 vol.2.

[119] L. Freda and G. Oriolo, "Frontier-based probabilistic strategies for sensor-based exploration," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 3881–3887.

[120] Ben Tribelhorn and Zachary Dodds, "Evaluating the roomba: A low-cost, ubiquitous platform for robotics research and education," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1393–1399.

[121] Michael K. Reed and Peter K. Allen, "Constraint-based sensor planning for scene modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1460–1467, Dec. 2000.

[122] María Teresa Lozano Albalate, Michel Devy, and José Miguel Sanchiz Martí, "Perception planning for an exploration task of a 3d environment," in *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3-Volume 3*. IEEE Computer Society, 2002, p. 30704.

[123] Kai M Wurm, Cyrill Stachniss, and Wolfram Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1160–1165.

[124] Howie Choset and Joel Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[125] Dirk Holz, Nicola Basilico, Francesco Amigoni, and Sven Behnke, "Evaluating the efficiency of frontier-based exploration strategies," *ISR/ROBOTIK 2010*, 2010.

[126] JG Rogers and Henrik I Christensen, "Robot planning with a semantic map," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2239–2244.

[127] Ran Zhao, Daniel Sidobre, and Wuwei He, "Online via-points trajectory generation for reactive manipulations," in *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*. IEEE, 2014, pp. 1243–1248.

[128] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 4, pp. 509 –522, apr 2002.

[129] D.M. Gavrila and V. Philomin, "Real-time object detection for "smart" vehicles," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999, vol. 1, pp. 87 –93 vol.1.

[130] R. Salakhutdinov, A. Torralba, and J. Tenenbaum, "Learning to share visual appearance for multiclass object detection," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, june 2011, pp. 1481 –1488.

[131] Paul Viola and Michael Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, 2001, vol. 1, pp. I–511.

[132] G. Barequet and S. Har-Peled, "Efficiently approximating the minimum-volume bounding box of a point set in three dimensions," *J. Algorithms*, vol. 38, pp. 91–109, 2001.

[133] Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip HS Torr, "Graph cut based inference with co-occurrence statistics," in *Computer Vision–ECCV 2010*, pp. 239–253. Springer, 2010.

[134] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie, "Objects in context," in *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*. IEEE, 2007, pp. 1–8.

[135] Google, "Google sets (dead link)," `http://labs.google.com/sets`, 2010.

[136] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014*, pp. 740–755. Springer, 2014.

[137] Alexander Kasper, R Jakel, and Rüdiger Dillmann, "Using spatial relations of objects in real world scenes for scene structuring and scene understanding," in *Advanced Robotics (ICAR), 2011 15th International Conference on*. IEEE, 2011, pp. 421–426.

[138] Tristram Southey and James J Little, "3d spatial relationships for improving object detection," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 140–147.

[139] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena, "Contextually guided semantic labeling and search for three-dimensional point clouds," *The International Journal of Robotics Research*, p. 0278364912461538, 2012.

[140] Aki Vehtari and Jouko Lampinen, "Bayesian mlp neural networks for image analysis," *Pattern Recognition Letters*, vol. 21, no. 13, pp. 1183–1191, 2000.

[141] Hilary Buxton and Shaogang Gong, "Visual surveillance in a dynamic and uncertain world," *Artificial Intelligence*, vol. 78, no. 1, pp. 431–459, 1995.

[142] Matthew Richardson and Pedro Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[143] Robert Crane and Luke K McDowell, "Evaluating markov logic networks for collective classification," in *Proceedings of the 9th MLG Workshop at the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.

[144] David Fernández, Simone Marinai, Josep Lladós, and Alicia Fornés, "Contextual word spotting in historical manuscripts using markov logic networks," in *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*. ACM, 2013, pp. 36–43.

[145] Anton Chechetka, Denver Dash, and Matthai Philipose, "Relational learning for collective classification of entities in images.," in *Statistical Relational Artificial Intelligence*, 2010.

[146] Lauro Snidaro, Ingrid Visentini, Karna Bryan, and Gian Luca Foresti, "Markov logic networks for context integration and situation assessment in maritime domain," in *Information Fusion (FUSION), 2012 15th International Conference on*. IEEE, 2012, pp. 1534–1539.

[147] Young Chol Song, Henry Kautz, James Allen, Mary Swift, Yuncheng Li, Jiebo Luo, and Ce Zhang, "A markov logic framework for recognizing complex events from multimodal data," in *Proceedings of the 15th ACM on International conference on multimodal interaction*. ACM, 2013, pp. 141–148.

[148] Luc De Raedt and Luc Dehaspe, "Clausal discovery," *Machine Learning*, vol. 26, no. 2-3, pp. 99–146, 1997.

[149] Stanley Kok and Pedro Domingos, "Learning the structure of markov logic networks," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 441–448.

[150] Parag Singla and Pedro Domingos, "Discriminative training of markov logic networks," in *AAAI*, 2005, vol. 5, pp. 868–873.

[151] Ross B Girshick, Pedro F Felzenszwalb, and D McAllester, "Discriminatively trained deformable part models, release 5," 2012.

[152] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[153] Gregory Griffin, Alex Holub, and Pietro Perona, "Caltech-256 object category dataset," 2007.

[154] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[155] Xavier Broquere, Daniel Sidobre, and Ignacio Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 2808–2813.

[156] Günter Schreiber, Andreas Stemmer, and Rainer Bischoff, "The fast research interface for the kuka lightweight robot," in *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010)*, 2010.

[157] Emanuele Magrini, Fabrizio Flacco, and Alessandro De Luca, "Estimation of contact forces using a virtual force sensor," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 2126–2133.

[158] Xavier Broquere, Daniel Sidobre, and Khoi Nguyen, "From motion planning to trajectory control with bounded jerk for service manipulator robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4505–4510.

[159] "Reflexxes," `http://www.reflexxes.ws/`, 2015.

[160] David Marr and Ellen Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[161] Eric Woods, Paul Mason, and Mark Billinghurst, "Magicmouse: an inexpensive 6-degree-of-freedom mouse," in *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. ACM, 2003, pp. 285–286.

[162] Séverin Lemaignan, Raquel Ros, and Rachid Alami, "Dialogue in situated environments: A symbolic approach to perspective-aware grounding, clarification and reasoning for robot," in *Robotics, Science and Systems, Grounding Human-Robot Dialog for Spatial Tasks workshop*, 2011, p. 1.

[163] Risto Koiva, Robert Haschke, and Helge Ritter, "Development of an intelligent object for grasp and manipulation research," in *Advanced Robotics (ICAR), 2011 15th International Conference on*. IEEE, 2011, pp. 204–210.

[164] Yuexing Han, Yasushi Sumi, Yoshio Matsumoto, and Noriaki Ando, "Acquisition of object pose from barcode for robot manipulation," in *Simulation, Modeling, and Programming for Autonomous Robots*, pp. 299–310. Springer, 2012.

[165] Shengyong Chen, Youfu Li, and Ngai Ming Kwok, "Active vision in robotic systems: A survey of recent developments," *The International Journal of Robotics Research*, 2011.

Résumé

Cette thèse traite des problèmes de modélisation, reconnaissance, localisation et utilisation du contexte pour la manipulation d'objets par un robot. Le processus de modélisation se divise en quatre composantes : le système réel, les données capteurs, les propriétés à reproduire et le modèle. En spécifiant chacune des ces composantes, il est possible de définir un processus de modélisation adapté au problème présent, la manipulation d'objets par un robot. Cette analyse mène à l'adoption des descriptor de texture locaux pour la modélisation. La modélisation basées sur des descripteurs de texture locaux a été abordé dans de nombreux travaux traitant de structure par le mouvement (SfM) ou de cartographie et localisation simultanée (SLAM). Les méthodes existantes incluent Bundler, Roboearth et 123DCatch. Pourtant, aucune de ces méthode n'a recueilli le consensus. En effet, l'implémentation d'une approche similaire montre que ces outils sont difficiles d'utilisation même pour des utilisateurs experts et qu'ils produisent des modèles d'une haute complexité.

Cette complexité est utile pour fournir un modèle robuste aux variations de point de vue. Il existe deux façons pour un modèle d'être robuste : avec le paradigme des vues multiple ou celui des descripteurs forts. Dans le paradigme des vues multiples, le modèle est construit à partir d'un grand nombre de points de vues de l'objet. Le paradigme des descripteurs forts compte sur des descripteurs résistants aux changements de points de vue. Les expériences réalisées montrent que des descripteurs forts permettent d'utiliser un faible nombre de vues, ce qui résulte en un modèle simple. Ces modèles simples n'incluent pas tout les point de vus existants mais les angles morts peuvent être compensés par le fait que le robot est mobile et peut adopter plusieurs points de vue.

En se basant sur des modèles simples, il est possible de définir des méthodes de modélisation basées sur des images seules, qui peuvent être récupérées depuis Internet. A titre d'illustration, à partir d'un nom de produit, il est possible de récupérer des manière totalement automatique des images depuis des magasins en ligne et de modéliser puis localiser les objets désirés.

Même avec une modélisation plus simple, dans des cas réel ou de nombreux objets doivent être pris en compte, il se pose des problèmes de stockage et traitement d'une telle masse de données. Cela se décompose en un problème de complexité, il faut traiter de nombreux modèles rapidement, et un problème d'ambiguité, des modèles peuvent se ressembler. L'impact de ces deux problèmes peut être réduit en utilisant l'information contextuelle. Le contexte est toute information non issue des l'objet lui même et qui aide a la reconnaissance. Ici deux types de context sont abordés : le lieu et les objets environnants.

Certains objets se trouvent dans certains endroits particuliers. En connaissant ces liens lieu/objet, il est possible de réduire la liste des objets candidats pouvant apparaître dans un lieu donné. Par ailleurs l'apprentissage du lien lieu/objet peut être fait automatiquement par un robot en modélisant puis explorant un environnement. L'information appris peut alors être fusionnée avec l'information visuelle courante pour améliorer la reconnaissance.

Dans les cas des objets environnants, un objet peut souvent apparaître au cotés d'autres objets, par exemple une souris et un clavier. En connaissant la fréquence d'apparition d'un objet avec d'autres objets, il est possible de réduire la liste des candidats lors de la reconnaissance. L'utilisation d'un Reseau de Markov Logique est particulièrement adaptée à la fusion de ce type de données.

Cette thèse montre la synergie de la robotique et du context pour la modélisation, reconnaissance et localisation d'objets.

Mots-clés: Modelisation, Reconnaissance, Localisation, Contexte, Cooccurrence d'objets, Reseau Logiques de Markov, Structure par le Mouvement, SLAM, Descripteurs de texture, Geometrie Multivue.

Abstract

This Thesis addresses the modelling, recognition, localisation and use of context for objects manipulation by a robot.

We start by presenting the modelling process and its components: the real system, the sensors' data, the properties to reproduce and the model. We show how, by specifying each of them, one can define a modelling process adapted to the problem at hand, namely object manipulation by a robot. This analysis leads us to the adoption of local textured descriptors for object modelling.

Modelling with local textured descriptors is not a new concept, it is the subject of many Structure from Motion (SfM) or Simultaneous Localisation and Mapping (SLAM) works. Existing methods include bundler, roboearth modeller and 123DCatch. Still, no methods has gained widespread adoption. By implementing a similar approach, we show that their are hard to use even for expert users and produce highly complex models.

Such complex techniques are necessary to guaranty the robustness of the model to view point change. There are two ways to handle the problem: the multiple views paradigm and the robust features paradigm. The multiple views paradigm advocate in favour of using a large number of views of the object. The robust feature paradigm rely on robust features able to resist large view point changes. We present a set of experiments to provide an insight into the right balance between both. By varying the number of views and using different features we show that small and fast models can provide robustness to view point changes up to bounded blind spots which can be handled by robotic means.

We propose four different methods to build simple models from images only, with as few a priori information as possible. The first one applies to planar or piecewise planar objects and rely on homographies for localisation. The second approach is applicable to objects with simple geometry, such as cylinders or spheres, but requires many measures on the object. The third method requires the use of a calibrated 3D sensor but no additional information. The fourth technique doesn't need a priori information at all. We apply this last method to autonomous grocery objects modelling. From images automatically retrieved from a grocery store website, we build a model which allows recognition and localisation for tracking.

Even using light models, real situations ask for numerous object models to be stored and processed. This pose the problems of complexity, processing multiple models quickly, and ambiguity, distinguishing similar objects. We propose to solve both problems by using contextual information. Contextual information is any information helping the recognition which is not directly provided by sensors. We focus on two contextual cues: the place and the surrounding objects.

Some objects are mainly found in some particular places. By knowing the current place, one can restrict the number of possible identities for a given object. We propose a method to autonomously explore a previously labelled environment and establish a correspondence between objects and places. Then this information can be used in a cascade combining simple visual descriptors and context. This experiment show that, for some objects, recognition can be achieved with as few as two simple features and the location as context.

The objects surrounding a given object can also be used as context. Objects like a keyboard, a mouse and a monitor are often close together. We use qualitative spatial descriptors to describe the position of objects with respect to their neighbours. Using a Markov Logic Network, we learn patterns in objects disposition. This information can then be used to recognise an object when surrounding objects are already identified.

This Thesis stresses the good match between robotics, context and objects recognition.

Keywords: Modelling, Recognition, Localisation, Context, Objects Co-occurrence, Markov Logic Network, Structure from Motion, SLAM, Texture Descriptors, Multiple View Geometry.