Object persistence in 3D for home robotics

by

Parnian Alimi

BSc. Computer Engineering, Sharif University of Technology, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

April 2012

Abstract

This document presents an interactive pipeline for collecting, labelling and re-recognizing movable objects in home-like environments. Inspired by the fact that a human child learns about a movable object by observing it moving from time to time, and memorizes its name once a name is associated with it, we have designed an object persistence system which performs similar tasks based on change detection. We utilize 3D *registration* and *change detection* systems in order to distinguish foreground from the background. Focusing on the dynamic objects (from different vantage points) and interactively asking the user for labels, endows our system with a database of labeled object segments, which further is used for *multi-view instance recognition*. We have expanded the temporal interval logic to 3D bounding boxes in order to aggregate regions that contain foreground dynamic objects, and simultaneously update our model of the background. The object segments are extracted by removing the background. Finally the objects are matched to the existing database of objects and if no match is present the user will be prompted to provide a label. To demonstrate the capabilities of our system, an inexpensive RGB-D sensor (Kinect) is used to collect 3D point clouds. Results show that for tabletop scenes, our approach is able to effectively separate object regions from background, and that objects can be successfully modelled and recognized during system operation.

Table of Contents

List of Tables

List of Figures

Acknowledgments

First and foremost I like to thank my supervisor, James Little, for his guidance and support to explore the research direction that I very much enjoyed. His emphasis on independent research as well as his willingness to share insights was an invaluable asset for my studies. I would also like to thank David Lowe for reading this thesis and providing valuable comments and edits.

I like to thank David Meger for brainstorming with me to solve various pieces of this puzzle. His expert knowledge and guidance saved me a great deal of time and effort. I also like to thank Pooja Viswanathan who motivated me by sharing her view and experience with assistive technology.

To Marzieh Razavi and Yasaman Sefidgar, my kind friends, for always being there for me.

At last, my highest gratitude goes to my parents, Susan and Reza, who always believed in me, and to my cheerful sisters, Mehregan and Mahshid. Without them I could not stand where I am today.

Chapter 1

Introduction

1.1 Motivation

The research conducted in this thesis is motivated by two applications which at first glance have different concerns but can share many components: *assistive* and *domestic* robotics.

The urge to advance *assistive technology* comes from the fact that there is an increasing population of older adults whilst nursing resources are limited. This application possesses its own set of useful tasks depending on the target users. For instance, given the fact that dementia is a common disease in elderly, automated guidance instructions for everyday tasks was one of the early topics that has been studied. Driven by this motivation Hoey et al. [23] present a vision-based hand washing system which prompts the users with a frequency proportional to its expectation of their awareness. Navigation assistance and obstacle avoidance are also two areas where an automated system was shown to be helpful [59].

Inspired by these works, we recognized semantically aware systems to be beneficial for cognitively impaired users. Having an understanding of the type of surrounding environment, the objects and the agents operating in it are potentially helpful to make an informed decision about the user's needs and also to initiate a natural dialogue with him/her. One can imagine a number of tasks in which a semantically aware system (which for instance has a *scene understanding* or *object recognition* module) can replace the need for a caregiver. For example to answer, "where am I?" and "where is the book that I was reading yesterday?". Although there exist systems that can track the user's location, an isolated localization module will not be able to answer the first question in a human understandable terms – it only can output a metric measurement with respect to its local map. However a scene understanding system which discerns room types will be helpful. To answer the latter question, a system is needed that can identify and locate objects and has chronologically observed the user's interaction with it. In this thesis we have focused on answering the object recognition related questions and left scene understanding as a future direction.

From the *domestic robotics* perspective, objects are the focus of a home robot's interactions with the world. They are the items to be grasped, they ground the meaning of terms in interaction with humans, and they are necessary for nearly every meaningful task performed by a home robot. However, most modern robots lack the ability to detect, track, and reason about the large number of objects that are likely to occur in modern homes. This is due in part to the challenges in describing and detecting objects once they have been demonstrated to a system.

Two facts to consider for an object recognition module that performs in such applications are that first the set of objects are limited and environment specific and second the agent operates in such environment for a while. To take advantage of these constraints our proposed system collects and maintains a database of objects. In this thesis we have focused on the set of objects which are moved by an agent (either human or other robots) over a sequence of observations.

The core reason for taking this approach for applications that perform life-long tasks in a single environment is that the localization and mapping modules that most of these robots are equipped with provide a good resource to track changes and extract information with minimal human intervention.

1.2 Contributions

In this thesis, we use an existing recognition approach, and we focus on one aspect of existing recognition paradigms. Nearly all proposed methods have been based on a separate teaching phase where a user once instructs the robot about the labels and appearances of items by hand-annotation, pointing or verbal commands. This labour intensive process results in a static database of objects that does not adapt as new objects are introduced to the environment. Our main intuition is that, since home robots autonomously operate in a particular environment for a long time, they can take advantage of the information they gather over time and incrementally construct object models in collaboration with their user.

To this end, we describe an *object persistence* system based upon analyzing changes in the environment. These changes become seeds to query the users for object labels (e.g. "What is the name of this object that has moved?") Once labels have been given, the system updates its persistent object model and focuses its recognition effort on explaining the subsequent changes. This use of history and focusing of attention reduces the computation required to continuously track the position of a large number of objects, which is a primary practical limiting factor that currently restricts the number of objects simultaneously available to a system.

Beyond change detection, a key portion of our method is the verification and update of the semantic object labels used by humans to describe objects. We achieve this by grouping nearby changed regions into those likely to be explained by a single coherent object and subsequently, matching 3D sensory signature within each region to those of the objects previously known to the system.

1.3 Definitions

The key terms that have been used in this thesis are as follows:

Static background: That part of the environment that scarcely moves. For a typical home environment this includes: walls, counters, book shelves, etc.

Dynamic object: Also referred to as *interesting* objects, are the subset of objects that are movable by agents (humans or robots). We use this term to distinguish the segments of the scene that have the potential to move from the static scene. Examples include: books, dolls, boxes, phone, etc.

Bounding box: The smallest cube (for 3D point clouds) or rectangle (for 2D images) that contains all points or pixels which belong to an object.

Regions of interest: The subspace that has the potential to contain dynamic objects. This is usually abbreviated as ROI. Kitchen counter and coffee table are two examples of such regions.

1.4 Overview of the architecture

Our object recognition and data collection pipeline can be summarized in 4 steps:

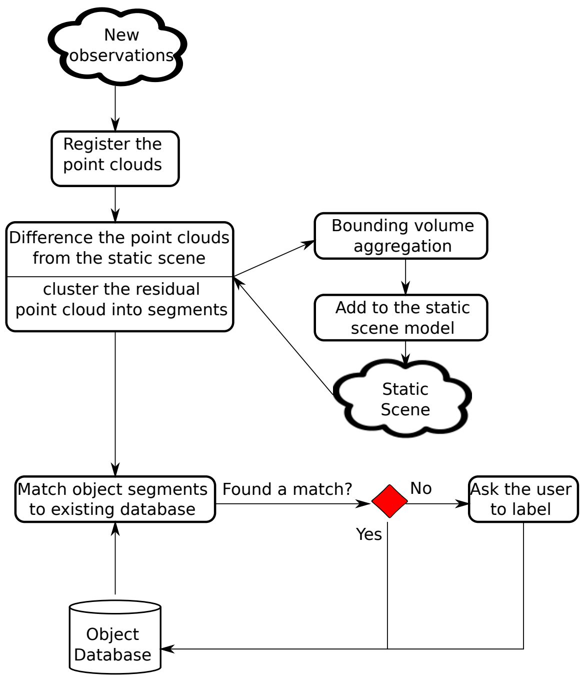

New observations
Register the point clouds
Difference the point clouds from the static scene
cluster the residual point cloud into segments
Bounding volume aggregation
Add to the static scene model
Static Scene
Match object segments to existing database
Found a match?
No
Yes
Ask the user to label
Object Database

Figure 1.1: Workflow of the proposed pipeline

1. Geometrically register the sensed data between visits, sequentially
2. Detect changes in the occupancy that may have occurred when objects moved
3. Group these changes to locate potential objects and update a background model of the scene
4. For object segments, attempt to associate a semantic label and add to the objects database

System components 1) and 2) act as an attention operator for our system, focusing later computation and allowing the system to quickly ignore large parts of the world that correspond to a static scene, which greatly improves efficiency. The result of subtraction is either a piece of static scene or a dynamic object.

Component 3) refers to the grouping of changes into various types: foreground or background. The farthest point that has ever been observed in each direction is considered as part of the static scene (i.e. background) and the system constantly updates its model. In some cases this simple reasoning may result in objects, in particular those that have previously been occluded, being categorized as background. However, this situation only persists as long as that object remains stationary, and once it is moved, our system can begin to assign it a label.

Component 4) involves update of the object database, recognition of known objects and user queries occasionally needed to obtain new labels. When a new object is located, the user is queried for its name. When a change is recognized as being an object that was previously known, the model for that object is updated. All of the previously seen objects that are unchanged can simply persist in the model, allowing for their use in tasks without additional perception effort.

The complete object persistence workflow is depicted in Figure 1.1.

Our system depends only on the reliable 3D data that is now available to many home robots, such as from an RGB-depth sensor like the Microsoft Kinect. This 3D data allows for highly accurate change detection, which correctly guides our approach only to update the state of objects that have moved. Also, appearance signatures derived from high resolution 3D sensory data are highly repeatable and this allows for successful tracking of the semantic labels of regions as objects are shifted, rotated, added and removed from a scene.

1.5 Organization

After this brief introduction, Chapter 2 discusses related work in semantic understanding and data collection areas. Next, Chapter 3 states the problem description, and is followed by Chapters 4 and 5 which describe our proposed attention and detection systems in more detail. Chapter 6 presents the set of experiments we conducted to investigate the strengths and shortcomings of our pipeline. This thesis is concluded by Chapter 7 which includes general discussions and suggestions for future work. Figure 1.2 illustrates the dependency ordering of the chapters.

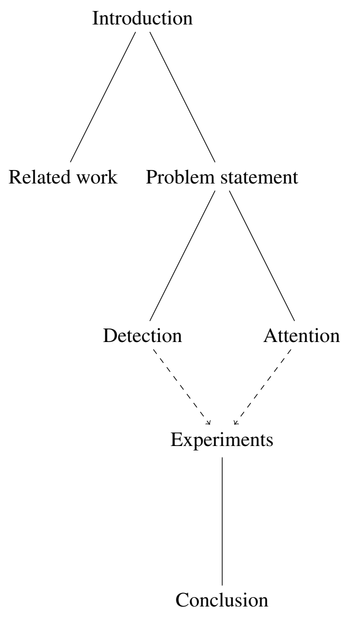


Figure 1.2: This thesis is best read in the order depicted in this figure.

Chapter 2

Related work

With an ultimate goal to have a robust object finding and recognition (semantically aware) system, this thesis is related to a number of research areas. A *scene understanding* module provides general information about the robots working environment. Next, to detect objects of interest (also referred to as movable objects in this thesis) in a cluttered environment an *attention operator* is needed. At last, the collected information about the objects need to be stored in a *database* for future use. The platform introduced in this thesis develops a procedure that combines registration and object detection modules for indoor object discovery.

2.1 Semantic understanding

2.1.1 Attention for object recognition

Object recognition in isolation has been investigated extensively in the literature requiring segmented and labeled data. A more advanced approach to object recognition can detect objects in context. To extend an object detector that detects isolated objects, the brute force algorithm is to search all the possible subspaces (image patches in 2D and 3D point cloud segments in 3D) for the object – so called template matching. The more effective approach would be to attend to only parts of the image for which, based on the feature keypoints or prior knowledge, we believe an object might be present.

The benefit of an attention operator for object recognition has previously been assessed for 2D images by Walther et al. [60] and Rutishauser et al. [50]. This also has been referred to as *object search* in the literature [6]. A reasoning system has been designed by Aydemir et al. [6] that starting from any arbitrary location and given both metric and semantic maps (built with a place categorization module), it will generate a sequence of actions to find a place that most probably contains the object. Once the object comes into its field of view it locates the object. In order to locate the object it uses its prior knowledge about where the objects can be placed, for example a cereal box is in the kitchen and most probably is placed on a table or a counter.

While Aydemir et al. [6] manually define the object and scene relations, Kollar and Roy [30] extract the *object-object* and *object-scene* relationships from the photo captions on Flicker website. The key idea behind this is that objects are located in predictable scenes given their type and the structure of various scenes. For instance, desks are located in offices with chairs, laptops, lamps, etc. Kollar and Roy [30] also introduce an algorithm to search for a novel query object (given a prediction about its possible location) on an optimal trajectory.

Marton et al. [37] categorize the perception task into *active* and *passive*. Active perception happens when the robot is actively searching the space for a set of specific objects. Passive perception refers to collecting position and identification information about the objects while performing various even unrelated tasks. If the output of perception module is the input for manipulation module, a 3D model of the interested object is needed. To manipulate objects a description of the shape is sufficient – identifying the object is not needed, e.g. the brand of the soda can. Their object recognition system learns each view of each object as a separate class; and at test time returns the object's class regardless of its view. They use the TF-IDF concept to make documents (object models) from words (visual features).

For applications where expert knowledge or web-based data is not applicable, an attention operator is needed to suggest areas that have potential to contain objects (also called regions of interest or ROI). Given this definition, attention for object recognition is a sub-system of a semantically aware intelligent system which obtains an environmental or world model [18, 45]. Rusu et al. [45] extract the planar surfaces to produce semantic information about ROI in a kitchen.

In a more generalized way Hager and Wegbreit [18] formulate object persistence with two key ideas: 1. Derive a world model from a sequence of visits. 2. Take advantage of physical constraints such as physical supporting relationships. They construct a *scene graph* which represents the relationship between the objects (scene dynamics) and the background as well as an ordering based on occlusion order between the objects. They precisely explain the procedure to maintain the scene graph after the changes in the scene occur. Their results confirm that the scene parsing for real scenes is more challenging than simulation – mainly due sensor limitations. In this thesis we only focused on real scenes.

2.1.2 Change tracking

The idea of tracking the changes in the scene in order to extract information in an unsupervised manner was previously explored by Herbst et al. [21, 22]. They devise a probabilistic measurement model for RGBD sensors in [22], plus a movement model for surface patches to determine which segments have been moved. Their experiments are focused on moved surfaces' segmentation quality and timing, in contrast to ours which are more focused on multi-view object recognition and data collection. Built upon this work in [21] they use Markov Random Field (MRF) to derive the probability of each surface patch being moved between frames.

With a similar motivation Southey and Little [53] perform object discovery by means of tracking the set of features that move rigidly between sequence of frames. They segment moving objects by combining appearance, shape and rigid motion properties. Beuter et al. [9] formulated the same problem by categorizing the space into static scene, moving entities and movable objects. As the observations are inputed to the system, it iteratively updates its model of the scene and tracks the moving entities. The motion model for moving entities is predicted with particle filtering. Later the static scene model is updated by excluding the moving entities from the current observation's point cloud and given the fact that the static scene is the farthest point observed at each direction. Also at each time step the current model of the static scene is used to extract parts of the moved objects.

2.1.3 Life long learning

Meger et al. [39] describe a method which uses image-based visual saliency to propose promising regions, previously learned object models to label those regions, and a SLAM framework to track detections over time. However, they do not describe how to update object models once they are acquired, which is a key component to avoid the computation needed to continuously re-recognize all objects. This concept is similar to so-called "lifelong mapping", which was first fully defined in [32].

Finally, an approach for representing and updating semantic information about an environment is given by Hawes et al. [20]. This work considers larger scale spatial information and focuses on selection of behaviours rather than focusing on updating a perceptual model, but we are inspired by several of their concepts.

A complete version of our proposed system is capable of navigating through the environment and obtaining a map which makes the comparison between observations possible. Further, it maintains a database of movable objects and their corresponding labels. In this respect the proposed system in [62] follows the same goal as ours. They put together mapping and object recognition modules in order to extract semantic information about the scene types which further is used for facilitating human-robot interaction through dialogue.

2.2 Data collection

2.2.1 Dataset properties

Generally, there are many parameters to measure the quality of a database, and there has been a lot of effort to build unbiased databases that are challenging enough to be useful in detecting realistic objects. According to Ponce et al. [42] a set of normalized images with a single object in the centre is not a realistic database since it can not distinguish between detectors that are robust to size, scale and rotation changes. Rather, images in a database should contain multiple object categories in a single image, partial occlusion and truncation (by image edges); moreover the size and orientation of the objects should vary between the images. While these criteria are designed to evaluate 2D image databases, all of them except size invariance could be used for building 3D point cloud databases.

Human photographed images tend to capture the *characteristic view* of the objects. For example a shoe is usually photographed from the side and hardly from the top or front. Since such views are not guaranteed due to occlusion, or are infeasible due the camera positioning, particularly in robotics applications, robot collected data is needed in addition to World Wide Web databases. Automatic data collection has been explored by a few groups.

2.2.2 Autonomous data collection

Perhaps the work by Meger et al. [38] is one of the earliest work that described an infrastructure to explore the environment and collect images of the objects from various views. Given an occupancy grid paths to interesting locations in the map are planned by A^* algorithm. In order to cover the environment, they employed a frontier based exploration. And by turning around an object of interest and covering different views of the object they increase the chance of capturing a characteristic view.

Jia et al. [25] have designed a utility function that encourages fitting more than one object in each image, as well as covering different views of an object instead of taking similar views. Moreover, rather than taking a foveal image as is done in [38], they compute the optimal distance to capture most parts of the object with highest possible resolution. At the end they plan the shortest possible path to the next position and orientation that maximizes utility. To find objects of interest, they train an original detector on an online dataset, and further will track that object from various views.

Sapp et al. [51] develop an infrastructure to capture sample images of the objects on a green background and later synthesize training images with various foregrounds and backgrounds. This approach eliminates the need for a human to perform the time consuming task of labelling. Their results proves that synthetic data augmented over real data improves precision and recall.

2.2.3 Sample 3D datasets

3D point cloud collection has been one of the focuses of robotics applications to provide input data for object and scene recognition systems. Databases usually contain either *3D video of scenes* or *3D object models*. The former is used in place recognition and loop closure for localization and mapping algorithms, and the latter is used as training data for 3D object recognition.

There are a number of 3D databases which are collected mainly to test SLAM algorithms. For example, AASS-loop and Hanover2 (available at [4]) have been used by Steder et al. [57] for place recognition and loop closure. Also recently there have been some efforts to build 3D point cloud databases from both indoors and outdoors environments. However most of the available systems does not rely on the type of the sensor that has acquired the data, the amount of noisiness and resolution of the point cloud depends on the sensor. Chapter 5 contains a detailed discussion on the quality of range sensors.

Perhaps the largest Web based 3D database is Google 3D Warehouse [2], which contains both indoor and outdoor objects. 3D synthetic object and scene models from Google 3D warehouse [2] has been used in [17, 33, 61] for 3D object recognition.

A turntable is used in many cases to acquire multi-view pictures/point clouds of objects [34]. This technique provides accurate pose labelling, and segmentation is robust given the fact that the object is above a planar surface. Moreover since the data is collected in a controlled environment, the relative position of the camera and the platform is given. To make the annotation less time consuming and efficient, in [34] a 3D model of the world is constructed further a set of segments, called "surfels", are suggested to human for annotation. When the labelling is finished, surfels are transformed back to their corresponding frames and a bounding box will be computed for each frame. We have adopted a similar approach to annotation.

Recently Janoch et al. [24] presented a database of images collected with Kinect sensor from a variety of real domestic and office like environments which has happened by crowd-sourcing.

2.3 Summary

In the field of home robotics there has been a huge body of work on each of the navigation, mapping, object search and recognition tasks. Still to the best of our knowledge – primarily due technical implementation difficulties – there has not been much progress on integrating these into an infrastructures to extract semantic information about the objects on-the-fly in a long run. Ideally a home robot equipped with vision sensors should have a visual SLAM system (such as the one proposed in [32]) to localize and map the surrounding environment, attend to regions which may contain everyday objects (given its prior knowledge about the environment or by extracting semantic meaning about the supporting surfaces), recognize the objects (after training a learner on a web based dataset or a previously gathered and labeled set of objects) and maintain an environment specific database. This thesis is an attempt to introduce such a system.

Chapter 3

Problem statement

Persistent object detection is a primary job in many home robotics tasks. This chapter presents a general overview of our goals, and the pipeline that we have designed to achieve them. The two following chapters, Chapter 4 and Chapter 5, provide details on each component of the pipeline as well as their role as part of the whole.

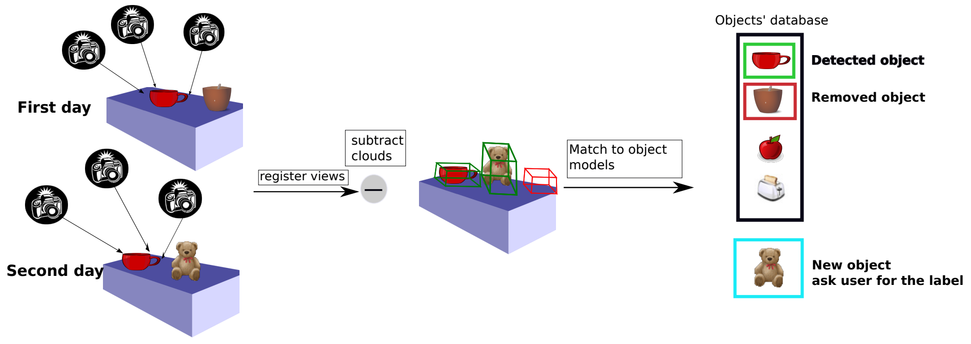


Figure 3.1: A diagram of the interactions between components in our persistent object detection method

In the big picture, our prototype visits a scene at several time slots and derives information about the objects in the scene based on the changes that it observes. Intuitively, after the human or other agents move the objects it can be inferred that the spaces occupied by those objects afford to contain interesting/dynamic objects. Over time, by combining these spaces, the robot can limit its search space to the combination of these spaces. Moreover from the collected object segments, it learns the object appearances and stores them in a database for future use. When none of the instances present in the database match with the newly observed segment, the system will query the user to provide a label. Figure 3.1 pictures the above scenario.

At a high level, the robot navigates through the scene, and at each *time step*, records the point cloud that it senses (which is a subset of the global model of the scene). Over the course of days or hours, the robot observes the same scene, over and over again, and in the meanwhile other agents relocate objects. For our experiments, we assume that the world is static during each visit, meaning that if there is a cat in the room, it does not move while the system records a full scan – in our experiments we have used a toy cat to make sure this constraint is satisfied. In other words, objects are allowed to move *between* observations, but not *during* an observation. In this chapter and the following two chapters, "time step" is used to address each shot that is taken during a single visit to a scene, and each visit is referred to as a single "observation".

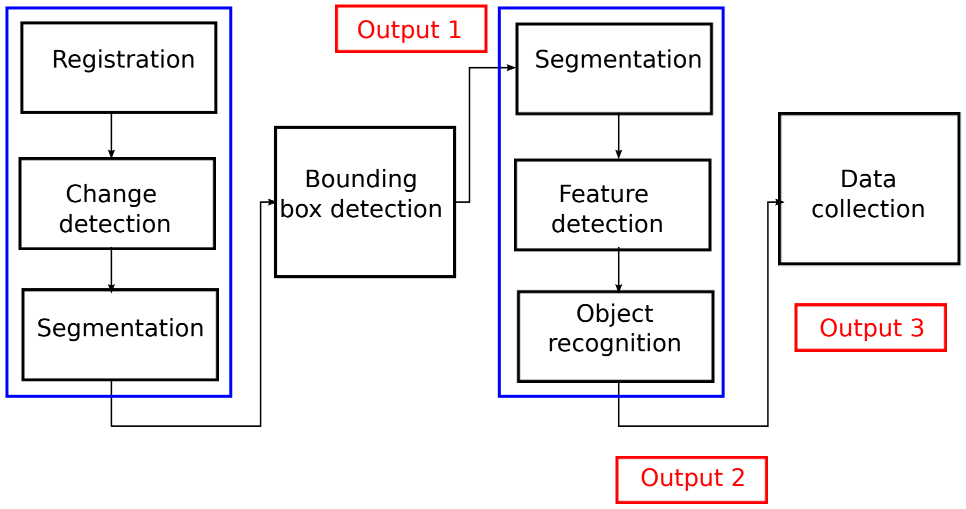


Figure 3.2: System overview

Figure 3.2 depicts the pipeline from another angle. It highlights the 3 outputs that the structure provides and can be used by various other modules: *regions of interest*, *object labels* and *a database of object instances*. *Regions of interest* are the places that afford to contain objects, therefore not only they can be used by object search module, but also can be used as working area for a manipulation module for example (pick and place manipulation operations). *Object labels* can be related to the user's vocal commands, (for example, "Fetch me my mug!"), and even further could be associated with other sources of data, for instance if the robot knows how to grasp a ball, a shoe, a glass, and a bottle, the object label guides it to choose the right set of instructions to execute; or if according to a user provided instruction set, it understands that a glass affords to contain liquids, when it is asked to fetch a glass of water, it can go, pour some water in the glass and bring it to the user. *Database of instances* not only can be used for further jobs by the same robot but also can be easily shared between robots operating in the same working environment, especially in nursing homes, where there are more than one user of this system and they share very similar appliances and utensils.

To produce these outputs from raw shots taken at each time step during several observations, at a high level, our system is composed of:

1. A geometric registration system that determines how the current sensor position differs from the position at which the scene was originally perceived

2. Geometric change detection that leverages the registration information to find objects that have likely moved during the period between observations (**Output 1**: regions of interest, at which changes occur)

3. A multi-view 3D object modelling approach that can be incrementally updated as new views of an object are located and which can assign semantic labels to each observed item in the world (**Output 2**: object labels)

4. A high-level object persistence reasoning module that combines the outputs of the previous three components to maintain a complete scene representation and add user information using queries when this is needed. (**Output 3:** database of object instances)

As depicted in Figure 3.1 system components 1) and 2) act as an attention operator for our system, focusing later computation and allowing the system to quickly ignore large parts of the world which are unchanged, which greatly improves efficiency. The system analyzes the changed regions with module 3) based on 3D feature descriptors and an efficient nearest-neighbours classifier. When a new object is located, the user is queried for its name. When a change is recognized as being an object that was previously known, the model for that object is updated. All of the previously seen objects that are unchanged can simply persist in the model, allowing for their use in tasks without additional perception effort. Finally, in some cases an object has simply been removed from the scene, so the corresponding object model is removed from the representation.

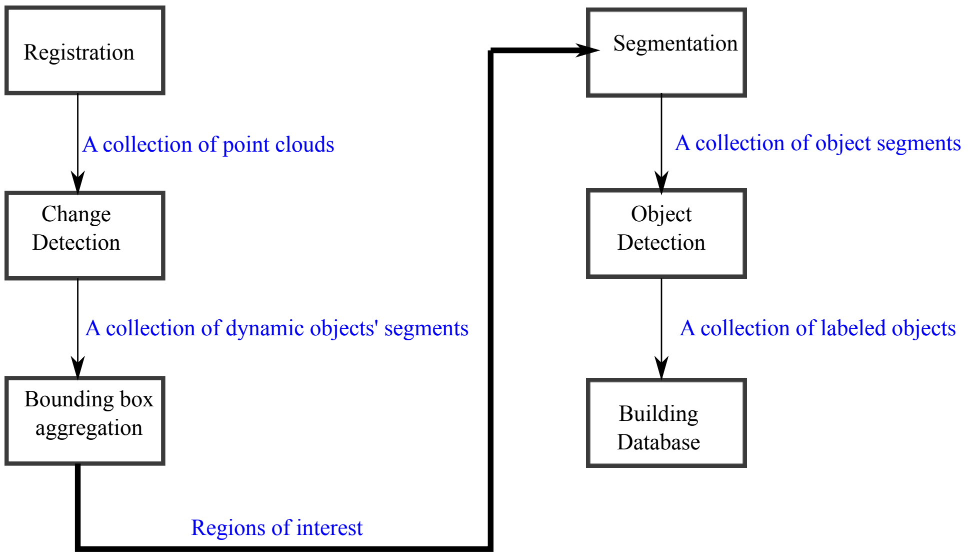

Registration
A collection of point clouds
Change Detection
A collection of dynamic objects' segments
Bounding box aggregation
Regions of interest
Segmentation
A collection of object segments
Object Detection
A collection of labeled objects
Building Database

Figure 3.3: System overview: the block representation, shows the flow of the pipeline as well as the output of each block that is inputed to the next block.

From the system design vantage, the process may split into two blocks: *attention operator* and *object detector*. Chapter 4 describes attention process that outputs regions of interest, and Chapter 5 illuminates the process of using regions of interest for object recognition.

Figure 3.3 sketches the relation between these blocks. The left and right blocks represent the attention operator and object recognition modules, accordingly. The attention operator is the cascade of *registration*, *change detection* and *bounding* *box aggregation* modules, which outputs a collection of sub-spaces of 3D space that, based on the collection of observations, afford to contain dynamic objects – *regions of interest*. The object recognition module extracts the set of points that are present in the regions of interest, segments them out, and passes them to object detector. If the recognition is successful, we have reached our goal – we know the object's location and identity. Otherwise, we will query the user. The corresponding chapters will illustrate the methods used for each component and challenges that have been encountered.

Chapter 4

System overview: attention

This chapter illustrates our attention operator in detail. First, we describe the registration approach that we took to build the 3D model of the world at each observation. Next, change detection and segmentation subsections explain the procedure that we take to extract object segments. At last, the bounding box analysis which unifies object bounding boxes to construct regions of interest, is described.

4.1 Registration

Our target applications are home robots and smart wheelchairs, which both are usually equipped with a localization module. Localization modules essentially take sensor data, and localize the vehicle in a precomputed map, or localize themselves while building the map – so called Simultaneous Localization and Mapping (SLAM) [54]. An accurate localization, with respect to the global frame/coordinates, empowers the robot to build a full 3D model of the world, which we use to represent the state of the world at each observation.

For the sake of our experiments, we have built a prototype that starting from a known position incrementally calculates the transformation between the robot's locations and orientations. This method is usually called "dead-reckoning" in the literature, primarily because it does not optimize the computed transformations as more information comes in, (i.e. it does not close the loops), and as a result accumulates error. We have limited our experiments to small parts of the room to avoid this issue.

There are three major reasons to collect more than one viewpoint of an object:

1. There is no guarantee that objects will be posed at the exact same orientation in future visits, nor one can assume that the robot would be able to navigate to the location where such vantage point will be possible. For example if the object will be placed on a shelf, without touching the object there is no way to sense the other side of it, or a shoe on the floor can not be observed from the side from a fixed camera location at the height of human eye.

2. An object will usually have more than one characteristic view. In particular, the objects that are not symmetric have more than one canonical pose, *e.g.* cats – their side view is very different from front or rear view (see Figure 4.1).

3. The object could be partially or completely occluded from one or more of the viewpoints, in previous observations.

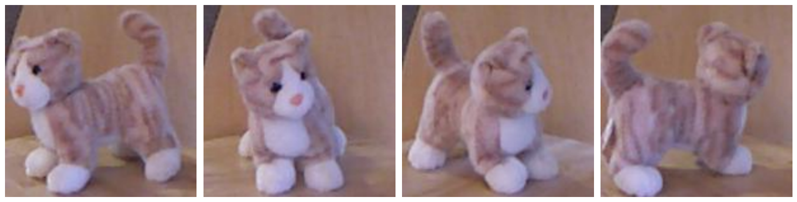


Figure 4.1: Cat is an example of objects with more than one canonical view

Registration is the problem of finding the transformation between the sensor data, at each time stamp, to the global/world coordinate system. The general problem of registration, independent of the dimensionality of the data (either 2D laser scans or 3D point clouds from stereo cameras), can be approached with 2 different methods: *point to point* matching or *feature* matching (i.e. correspondences). Both of these approaches calculate the transformation that aligns the overlapping areas in the frames. Generally, *point to point* matching approaches solve for the candidate transformation that aligns the biggest portion of the data. On the other side, methods based on *correspondences* suggest the transformation that aligns the data in a way that matched keypoints would be placed onto each other (since some sensor noise always exists, the point clouds would not be perfectly aligned).

Based on this definition, given the 3D location and orientation of the features and the correspondences between 2 point clouds (with the assumption that all the elements in the scene are rigid bodies) each correspondence plus the surface normals at the keypoints provides the 6 essential parameters, to transform the coordinate system of one cloud to the other. Since the correspondences can be noisy, an outlier removal technique, *e.g.* RANdom SAmple Consensus (RANSAC) [14], is run on the set of candidate transformation to find the one that most number of correspondences will vote for. Point to point matching techniques are usually dependent on the initialization, and on the other hand a noisy set of correspondences will result in an off transformation. Iterative Closest Point (ICP) [8] is perhaps the most well known point to point matching algorithm, but it fails in cases where the overlapping area is very small or the initial guess is not close enough to the goal transformation.

In our implementation, ICP is used for aligning partially overlapping point clouds. The algorithm iteratively recomputes the transformation (rotation and translation) between two point clouds by mapping the closest points. Depending on the size of point clouds, it can be slow, and like other gradient descent methods, convergence is not guaranteed. The termination criterion can be a threshold on the distance between the mapped point cloud and target cloud, or the number of iterations.

In our experiments we noticed that the more cluttered the scene gets, ICP outputs a closer transformation, mainly due the fact that the background/static scene is usually a combination of planar surfaces, e.g. desks and walls, for which most transformations (even a random transform) can result in a high matching score as long as a big portion of the point clouds overlap. Based on our initial experiments, since we move our prototype very slowly, ICP works the best in most cases. However for a clean scene, where there is no clutter, the scene is mostly contained of planar surfaces, for which ICP sticks in local optimum very quickly. For these cases a correspondence based approach seems to be more successful.

We also experimented with 2 state of the art feature detectors, for point cloud registration: Normal Aligned Radial Features (NARF), and Point Feature Histogram (PFH). To estimate a candidate transformation, we ran RANSAC on the correspondences that NARF or PFH offered. These feature detectors are usually coupled with a keypoint detector. NARF is computed on points in the scene that are placed on corners, and we computed PFH on the points that an extension of SIFT [36] keypoint detector, to 3D, offers.[1] We found transformation that are computed based on correspondences more powerful for plain scenes, compared to ICP. Plain scenes represent only the static part of the world (clear from dynamic objects), which usually are constituted from big planar surfaces (*e.g.* table tops, walls,...). In these cases a greedy method, with number of aligned point as utility function, fails right away. The rest of this section is an overview on NARF and its reported performance in previous work. Section 5.1 provides a description of PFH. Our initial investigations determined that PFH was a better candidate for this task, therefore we only used PFH in the experiments. Since our scenes are relatively cluttered, NARF fails to find stable correspondences between different views of the scene.

4.1.1 NARF

For many objects, boundaries are enough for successful recognition. For example a chair can be easily distinguished by its borders. Normal Aligned Radial Features (NARF), introduced by Steder et al. [58], is a local feature detector that picks border points as keypoints, and takes advantage of fast methods of extracting foreground from background in range images. Prior to this work, Steder et al. [55, 56] proposed a variant of Harris corner detector [19] to pick keypoints in range images. The descriptor for each keypoint was a 2D range image, that was captured by transforming the viewpoint to align with normal vector, in the opposite direction, which made the descriptor independent of observer's location. At the end to make the descriptor scale invariant, they would normalize the range values. These feature detection procedures were computationally expensive, moreover the keypoints could fall on the background instead of object border. To overcome these problems, instead of computing gradients on range image, they have proposed an algorithm to pick the points that are to the right, left, top or bottom of the borders, plus the points at which there is an abrupt change in principal curvature (to capture

[1]To expand 2D SIFT [36] keypoint detection to 3D, calculate difference of Gaussians at each scale, then find the local maxima, but instead of 2D Gaussians use 3D Gaussians. This is still performed on intensity but down sampling is performed with voxel grids, in 3D. More detailed description is available in PointClouds Library (PCL) [47].

shape properties). Moreover to compress the descriptor, they project a star shaped pattern on the normal aligned patches, around the keypoint, and only record the change in range value across the lines on the star.

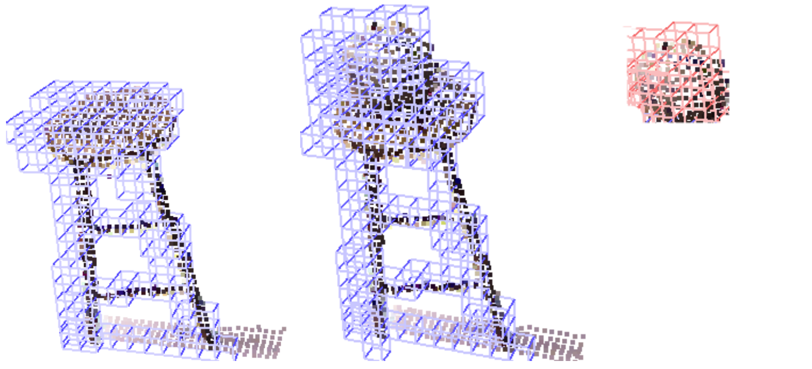
Picking borders as salient points breaks down in clutter, where the boundary of occluded area is not part of the object's boundary. In fact, a change in viewpoint will make these keypoints unstable. Unfortunately, the provided pictures of the scenes do not convince the reader that they have considered occlusion. In all the reported scenes, the objects are presented in isolation and often in foreground. The set of objects that have been used are limited to 7 objects with different sizes, but they are all highly symmetric.

On another note, in the applications where the robot maneuvers on a flat surface, it is not free to rotate about the roll axis (perpendicular to the image plane), therefore rotation invariance about roll axis is not a necessary property and in fact undesirably increases the search space. NARF is designed so that it can be used in either rotation variant or invariant modes. Steder et al. [58] report an extensive set of experiments on the keypoint's stability and the descriptor's recognition performance given 3D models of objects in the scene. But, unfortunately they only compare NARF with their previous detector [56] and not other 3D feature detectors.

The final remark is that this feature detector has been designed for place recognition and loop closure in SLAM from range data. In [57], NARF features are used to match the scans with a bag of words approach, and also to provide a 3D transformation between the frames. They have experimented on 4 different datasets including a flying Quadrotor robot (for which rotation about roll and pitch axes are allowed.). Although there are many hardcoded thresholds in their method, the results are very promising.

4.2 Change detection

A dynamic object is a rigid body that in contrast with background/static scene can be moved, removed or left untouched. For instance, in indoors scenes, they can be objects that a human carries (book, cup, bag, plate, etc.) or the objects that he/she can move (chair, computer mouse, computer monitor, etc.). The scope that each of these objects can be placed on can be relatively restricted (for example computer mouse would not be moved far away from the computer station) or unlimited (for example a bag can be placed anywhere).



(a) stool **(b)** ball placed on the stool **(c)** ball is new!

Figure 4.2: The three steps to detect the added points to the scene. 4.2a and 4.2b represent the scene before and after adding a ball. 4.2c highlights the voxels that contain the ball. Voxel grid fills all the space but for presentation clearance purposes, only the occupied voxels are drawn.

Our methodology considers a point cloud segment to be a dynamic object only when it is moved. Therefore, an object ought to be displaced before it can be noticed as a movable object. To make the distinction between static and dynamic parts of the scene we either need to obtain an observation from the clean scene or constantly build a model of the static background. In cases where the static scene (a clean observation) could not be captured in a single observation, the residual segments (from scene differencing) belong to either the static scene or dynamic objects. We consider the clusters that come out of occlusion (the regions with farthest observed distance to the viewer) to belong to the static scene, and add them to the current model of the static background.

From an implementation perspective, in order to detect changes in point clouds (all stated in the world coordinate system), we store the current model of the static background in a voxel grid, and then add the new point cloud to the voxel grid and count the voxels that only contain points from the new cloud, and those are parts of added objects. Figure 4.2 illustrates a very simple scenario.

4.3 Segmentation

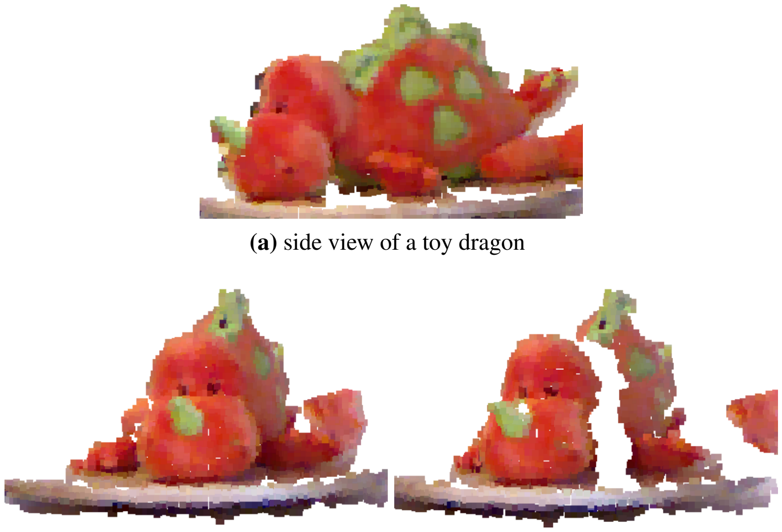
In our work we have used a heuristic method to clear the residual point cloud (after subtracting the two clouds), from noise and cluster the object segments: If a point has a closest distance less than a threshold to a cluster, it belongs to that cluster. With the propositional logic terminology, a point (p) belongs to a cluster (C) if and only if there is at least one other point (q) in that cluster which is closer than a threshold (δ) to that point (p):

$$p \in C \iff \exists q \in C : \|q - p\|_2 < \delta \tag{4.1}$$

Since in this thesis we focus on the point clouds that represent the real world, distance is defined as Euclidean distance. In our experiments we have used $3cm$ [2] as distance threshold. For details on the implementation please refer to Rusu [46] PhD thesis.

For three major reasons the change detection procedure explained in Section 4.2 does not output a single segment for an object. *First*, due to occlusion by other objects in the scene or self-occlusion, a single rigid body could be split into several pieces. (See Figure 4.3) In such cases, a part of the object could not be observed and/or it is divided into pieces. Detecting these situations given a single point cloud is not possible, but as more observations are made this issue can be resolved. Also, observing the object from another angle, if possible, will resolve this problem. *Second*, when the Regions Of Interest (ROI) only contains a portion of the object, for example, replacing a small object with a bigger object is a situation where ROI is smaller than the object and only contains a portion of the object. Section 4.4.3 provides more detail on various scenarios. Region growing is a reasonable solution for this problem. The part of the object that is within ROI can be used as seed and it can be grown to contain all the object. However since the boundary of the object cannot be calculated without any prior about its shape or size, there is no trivial approach to achieve this. Rusu et al. [45] derived a procedure to grow the regions in directions which their points have the smallest change in normal direction. As

[2]This value depends on the depth resolution of the sensor that is used for point cloud collection.


(a) side view of a toy dragon

(b) front view of a toy dragon **(c)** 4.3b rotated to the side

Figure 4.3: An example of self occlusion, which splits the object into several pieces.

one can expect, this results in growing the planar surfaces. Since the object shapes that we have investigated mostly are not flat, we can not employ such method for region growing. Also extending this method for arbitrary shapes does not look straightforward. In our frame work, as more observations are made, the ROI may get bigger and it may contain other parts of this dynamic object. Therefore, iteration over the observations is a solution to catch these cases. Once other segments of object are discovered, the database should be updated to contain the new data accordingly. *Third*, another issue arises when objects are touching each other or are closer than the δ threshold. In these cases objects will be clustered as a single body. Examples are provided in Chapter 6.

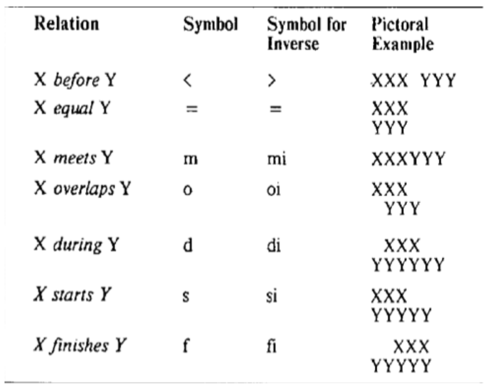
4.4 Bounding box analysis and reasoning

To characterize the location of the objects in the data it is common to estimate their shapes with convex hulls or axis aligned cubes. In this work we studied the space that dynamic objects occupy, the so called ROI. To derive the ROI from a number of object bounding boxes, the relative position of these boxes has to be analyzed. In this section we will demonstrate the logic behind aggregating 3D bounding boxes. To simplify the problem, we break the 3D reasoning into three 1D sub-problems and further unify the information. Other alternative was to unify bounding boxes into polygons. We extended the *Interval logic* since it was easy to generalize to 3D, and computing union was easier with this logic.

Section 4.4.1 is a brief introduction to *Interval Logic* (in 1D). Its goal is to establish all the possible relations between a set of intervals based on the transitivity rule. Next interval logic is extended to higher dimensions. Finally, various scenarios that might occur to the dynamic objects' bounding boxes are analyzed. In the rest of this section, the term "interval" is used for subsets of 1D space and "box" for 2 or 3 dimensional subspaces.

4.4.1 Interval logic

Interval Logic was first introduced by Allen [5]. It focusses on 1D intervals which could be the representation for actions occurring in a period of time. According to Interval Logic, any 2 intervals have 13 possible temporal relations – that are mutually exclusive: before, after, during, contains, overlaps, over-lapped-by, meets, met-by, starts, started by, finishes, finished-by, and equals. Figure 4.4 illustrates these relations.



Relation	Symbol	Symbol for Inverse	Pictoral Example
X *before* Y	<	>	XXX YYY
X *equal* Y	=	=	XXX YYY
X *meets* Y	m	mi	XXXYYY
X *overlaps* Y	o	oi	XXX YYY
X *during* Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Figure 4.4: The thirteen possible relationships between intervals, by Allen [5]. ©ACM 1983

Some algorithms have been designed to obtain the relationship between any two intervals in a set of intervals. One is to construct a network that has intervals as *nodes* and mutual relations as *edges*, and utilizing transitivity axiom, find the relation between two intervals by traversing the path between them [5].

4.4.2 3D bounding box aggregation

In this work we aim to expand an ROI by unifying overlapping object bounding boxes. To achieve this, the following algorithms have been proposed:

Algorithm 1 Growing region of interest

```
1: Grow-Bounding-Boxes (B1, B2)
2: $N \Leftarrow \emptyset$ {set of combined bounding boxes}
3: $B \Leftarrow \emptyset$ {set of B2 boxes that overlap with one or more boxes in B1}
4: for all b1 $\in$ B1 do
5:     $A \Leftarrow \emptyset$ {set of B2 boxes that overlap with b1}
6:     for all b2 $\in$ B2 do
7:         if overlap($b1$, $b2$) then
8:             Add $b2$ to $A$
9:         end if
10:    end for
11:    if A is Empty then
12:        Add $b1$ to $N$
13:    else
14:        Add the smallest box containing $A$ to $N$
15:        Add $A$ to $B$
16:    end if
17: end for
18: for all $b \in B2 \wedge b \notin B$ do
19:     Add $b$ to $N$
20: end for
```

Each box is presented uniquely by a pair of corners that are positioned diagonally (note that the bounding boxes are axis-aligned.). Among the 4 possible pairs of diagonal corners, the corners that have minimum and maximum values at every dimension were picked. Let's name them min and max corners. The two representing corners (min_{new} and max_{new}) for smallest box containing a set of boxes can be calculated from the min and max corners of each box. In this representation, relative position of the boxes can be determined, conveniently. The minimum values at each direction among the min corners will form the tuple representing the min_{new} corner and maximum values at each direction among the max corners will form the tuple representing the max_{new} corner.

To find the smallest box containing a set of boxes, the minimum and maximum of the 2 ends of the intervals will be found at each dimension.

Algorithm 2 Overlap(A, B) : To check if two 3D boxes overlap

```
1: $p \Leftarrow True$
2: **for all** dimensions **do**
3:     $a \Leftarrow A_{dimension}$
4:     $b \Leftarrow B_{dimension}$
5:     $p \Leftarrow p \wedge ((a\ during\ b) \vee (b\ during\ a) \vee (a\ contains\ b) \vee (b\ contains\ a) \vee (a\ overlaps\ b) \vee (a\ overLappedBy\ b) \vee (b\ overLappedBy\ a) \vee (a\ equals\ b))$
6: **end for**
7: **return** p
```

We assume that in each set, none of the boxes overlap. But for dimensions higher than 1, after aggregating two sets, the new set may contain regions that did not exist in any of the original boxes, as illustrated in figure 4.5.

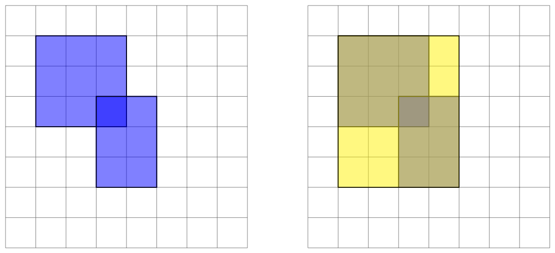


Figure 4.5: The new bounding box can contain regions that do not belong to either boxes. Left image: two overlapping boxes (in blue). Right image: the new box (in yellow) overlaid on old overlapping boxes.

Therefore another pass through the set is required to check if the newly generated boxes overlap; and combine them in case of overlap.

4.4.3 Various replacement and object positioning scenarios

Dynamic objects can be moved in many different ways: They can be added or removed from the scene, they can be shifted or rotated and they can occlude each other. Our goal is to detect changes in a scene based on subtracting the point clouds at consecutive time frames. In this section we will analyze various situations and the policy developed to derive information from these events.

To simplify the explanation, here we explain the scenarios in 2D and for primary shapes, *e.g.* circles, triangles and rectangles. We believe this analysis is extendable to arbitrary shaped objects, and to support that assumption, the experiments have been conducted with different shaped objects used in everyday life.

In all the figures in this section the leftmost, middle and rightmost pictures illustrate the first frame, the second frame and the difference between the 2 frames, respectively. Also the red, blue and purple colours determine the removed, added, and overlapping areas, accordingly.

Replacement: Based on the size and shape of the objects various scenarios can occur. Basic scenarios are depicted in Figure 4.6 and 4.8. The next case is if the object is replaced with an object with similar size. As illustrated in Figure 4.7 by subtracting the second image from the first one the result will be the 3 blue regions for which a segmentation algorithm, based on Euclidean clustering (defined in Section 4.3), will fail to recognize a single shape.

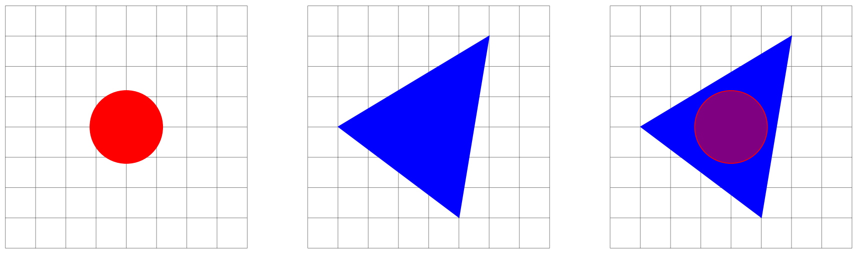


Figure 4.6: Replacing an object with a bigger object

The typical range sensors that have been introduced to the moment only sense the exterior of the objects; the resulting point clouds does not contain points to represent the interior of the objects. As a result, for instance a straw and a cable will both be observed as cylinders. Figure 4.9 illustrates a case in which a circular

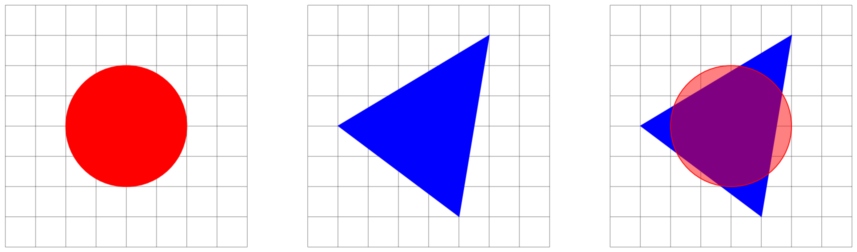


Figure 4.7: Replacing an object with a similar size object

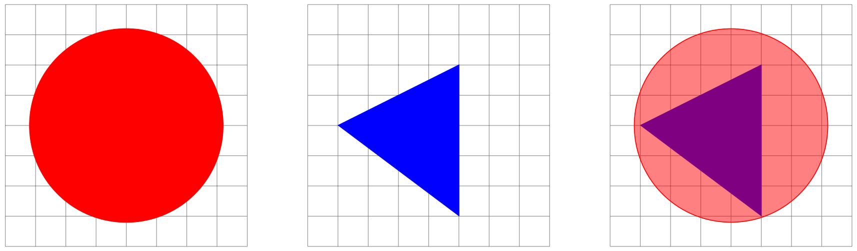


Figure 4.8: Replacing an object with a smaller object

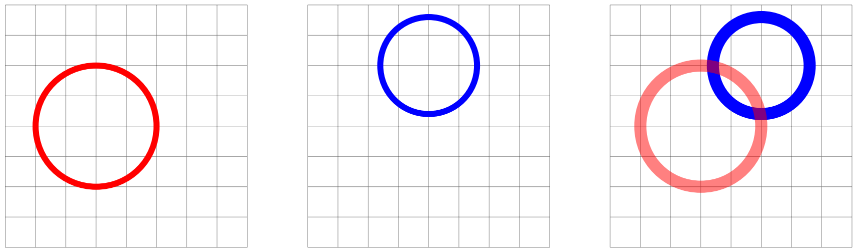


Figure 4.9: The point clouds only represent the exterior of the objects. This figure illustrates the case where an object is replaced with another one, and as it is evident their point clouds only overlap on 2 points in 2D (and by extension a 1D trajectory for 3D).

object is replaced with another one.

Occlusion reasoning

In real world applications, objects occlude each other. In other words, more than one point's projection from 3D to camera plane is a particular pixel in the image. The camera will only capture the closest point and all the other points will be hidden. When the object that is occluding parts of the scene, is removed, three scenarios are possible as illustrated in figure 4.10. This event reveals a static part of the scene and possibly all or parts of a dynamic object. Since no evidence is available about which parts are dynamic we treat the whole region as a static scene and wait for future observations to make the decision whether it is an interesting region or not.

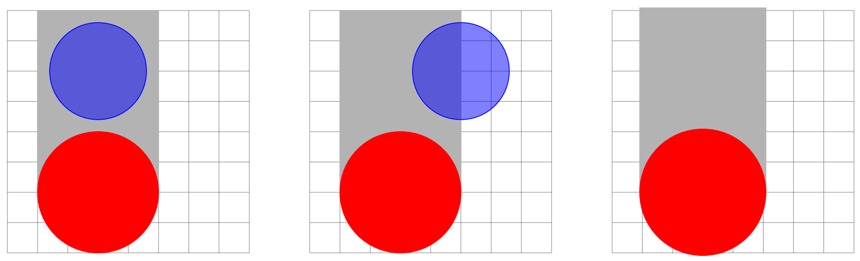


Figure 4.10: Occlusion reasoning: assume the camera is located bellow and is looking upwards. The grey region is the occluded area. The Blue circle is a dynamic object.

Chapter 5

System overview: detection

In this work we have developed an infrastructure to accumulatively maintain a database of object instances, that appear in the working environment of the robot, and constantly recall and record the position of the objects that previously have been observed. Our ultimate goal is to have a mechanism in which a human user names each object once and then the system must recognize that object as it moves into new locations and is seen from different vantage points. We describe objects based on their shape and visual appearances.

The sub-system that is introduced in this chapter takes the object segments (output of *attention operator*) as input, and maintains a database, and in a parallel process, outputs the label for the segments that successfully match to one of the instances in the database. In the following sections, first, Section 5.1 is an overview on the 3D feature detectors that we have used in our object recognition or registration modules. It is followed by an overview of the object recognition system. Section 5.2 describes the mechanism for maintaining the database, and tracking the objects, over time. At last Section 5.3 discusses depth sensors that can be used to acquire 3D point clouds, that the system takes as input.

5.1 3D features and multi view object detection

In this section we discuss the properties of various feature detectors that we have investigated. The list of feature detectors that we have studied deals with two main challenges in object recognition: *viewpoint variation* and *occlusion*. We have explored various scenarios that involve visiting an object from new viewpoints to examine each feature detector's robustness to this factor. For instance, the Viewpoint Feature Histogram (VFH), by Rusu et al. [49], calculates a descriptor that encodes viewpoint specific characteristics, while other feature detectors that have been used are viewpoint independent. Since occlusion and clutter also affect object appearances we have explored the feature detectors' ability to describe objects from partial views. Shape descriptors operate either on parts or the whole object. In this respect, spin images, by Johnson and Hebert [27], PFH/FPFH, by Rusu et al. [44, 48], explain the object's shape locally at the keypoints, whereas VOSCH, by Kanezaki et al. [28], and VFH explain it in its entirety. Local features are expected to be more robust to occlusion, compared to global features which require the whole object to be present in order to be successful.

Many other 3D shape detectors have been introduced, which are not discussed in this section. For example, Darbandi et al. [12] describe a shape descriptor which encodes flatness, orientation and convexity (FOC) of the local surface in a histogram. Endres et al. [13] introduced a variant of spin images, which stores the angular distance between the surface normals and the oriented point at the origin of the image instead of storing their positions. They also focus on generic object detection as opposed to surface matching.

We have carefully selected these set of features in order to cover both local and global features, viewpoint variance and invariance, rotation variance and invariance, appearance, and shape descriptions. In all object matching cases, we have used a leave-one-out approach, in other words, we train on all of the objects except the query object. The implementation, for all the used features, is available in Point Clouds Library (PCL) [47].[1]

5.1.1 PFH family

The PFH family of 3D feature detectors are developed by Rusu et al. [44, 48, 49]. FPFH [48] was proposed after PFH [44] and is its faster variant – since it only looks at a close neighbourhood of each keypoints for shape properties. Further

[1]http://pointclouds.org/

they introduced VFH which builds a histogram for the whole object out of FPFH descriptors and also incorporates viewpoint information into it. Therefore it detects the viewpoint from which it observes the object – if it has seen it before. This is particularly useful for applications that need to estimate object's orientation as well as its identity, *e.g.* manipulation.

For a patch around the query point, PFH estimates the relative direction of normals for every combination of 2 points on the surface, which is $O(n^2)$ if there are n points on the surface. FPFH shrinks the computation space by only considering K nearest neighbours of each point on the surface, which makes it $O(K*n)$. VFH, on the other hand, only computes the relative angle between viewing direction, drawn at the centre, and all the surface normals, only once, which therefore is $O(n)$ [49].

PFH

PFH [44] is a local feature descriptor which encodes the 3D structure around a query point. The descriptor for each point is a histogram made by concatenating 4 different characteristics of combinations of two points selected within a radius distance of the query point. The characteristics that form the descriptor are the relative angle between the point normals, the distance between the points, and the relative angle between the line connecting the points and the Darboux coordinates fixed at one of the points [44]. At the next step each feature histogram is classified as a shape primitive: plane, sphere, cylinder, cone, torus, edge or corner. To learn these shapes, Rusu et al. [44] train a classifier on a synthetically generated set of primitive shapes. Further, the set of histograms for each shape is divided into k clusters. The k clusters computed in this regime will be treated as most discriminative features and by learning only these features, the computation time will decrease.

In [43] the correspondence between PFH descriptors is used to register two point clouds. The computed transform is used to initialize the Iterative Closest Point (ICP) algorithm.

FPFH

FPFH [48] was proposed as an evolved version of PFH mainly to improve computation time and its descriptiveness. According to [48] experiments revealed that the distance between neighbours which was originally part of the descriptor [44] does not improve the robustness, because in 2.5 D data the distance between the neighbouring points depends on the distance to the camera. In [49], it is pointed out that GFPFH (Global FPFH) – which essentially constructs a global descriptor from the classification results of some FPFH descriptors – is not a powerful object detector, caused by its sensitivity to false local detections. FPFH is invariant to scale and viewpoint change.

VFH

VFH [49] computes FPFH for the entire object segment, with the difference that view direction is also incorporated with normals. Therefore, it has two components: the relative angle between *viewpoint direction* and the normal at each point, plus the relative angle between the viewpoint direction at the centroid of the object and each of the surface normals.

It is also designed to tolerate depth noise. In order to compute VFH on an object the point cloud must only contain the object segment – objects can not be detected in a scene. They report the recognition results for a large number of views of objects placed on a turntable. However the results are very promising for object recognition and pose estimation, compared to spin images, it only has been tested in light clutter. We designed experiments to indicate which one is more powerful to detect objects, in clutter.

For object recognition in particular, we form VFH descriptors [49] for each view of each object. Each view v_i of each object is then represented as a 308 dimensional feature vector $f_i = f(v_i)$ which describes the 3D appearance. Object recognition occurs by matching the features extracted from changed regions in new views ($f_n = f(v_n)$) to the entire set of vectors previously extracted within the set of known persistent objects. The nearest neighbour search, $argmin_i d(f_i, f_j)$ is performed efficiently using the Fast Library for Approximate Nearest Neighbours (FLANN) package provided by Muja and Lowe [41]. With this approach we can practically compare changed regions to many views of many known object models in real time.

5.1.2 VOSCH

Kanezaki et al. [28] develop a detector that combines the descriptors, which each operate on either shape or appearance of the objects. They essentially merge *Global Radius-based Surface Descriptor* (GRSD) [7] and *Circular Color Cubic Higher-order Local Auto-Correlation* descriptor (C3-HLAC) [29], and call the resulting histogram "Voxelized Shape and Color Histograms" (VOSCH).

C3-HLAC encodes the correlation of RGB (red green blue) values, between the voxels of a $3\times3\times3$ grid, high-dimensional vector. To compute GRSD, voxels are categorized into, free space, plane, cylinder, sphere, rim and noise, based on the principle curvatures of the voxel surfaces. Then, the change in surface type between neighbouring voxels contribute to the histogram that describes the object's shape. Finally these two histograms will be concatenated and normalized.

To resolve global features' problem in detecting partially visible objects, they divide the space into overlapping boxes, and further compute the feature histogram for each of the subspaces. For a few cylindric shaped objects with different colours, that we have in our database, we expect to see VOSCH outperform other detectors. Results in [28] are presented for data collected with the Kinect sensor.

Kanezaki et al. [28] report object recognition performance for a database composed of synthetic and real data with different colours, and under various lighting conditions. The recognition results look very promising, close to perfect detection. However it is not clear how much each of GRSD and C3-HLAC have contributed to recognition, in fact since their test data only varied in texture (different brands of milk boxes) the contribution of shape descriptor could have been very low.

VOSCH histograms are viewpoint independent, so we only compute them for the complete model of each object, instead of the partial segments, from each view. Therefore, pose estimation is not possible. In our experiments VOSCH histograms have 137 bins. To detect objects, we have used the same nearest neighbour approach as explained in previous section.

5.1.3 Spin images

Spin images [27] describe the shape at each point on the surface, locally. Therefore, to perform surface matching or object detection, spin-images on two surfaces should match at a number of points. Matching points locally, is a solution to clutter and occlusion.

The spin image at each point is defined as the 2D projection of the local surface into the cylindrical coordinate system. To consider a point to be in the local neighbourhood of the query point, it must be closer than a *supporting distance*. Plus its normal must make an angle less than *supporting angle* to the normal at query point. The first property preserves the locality of the shape descriptor and the second constraint filters abrupt changes in normal which usually correspond to edges – where surface normal estimation is not stable.

For surface matching Johnson and Hebert [27] propose to compute spin-images at every vertex on the surface mesh or on a uniformly sampled subset. Each query point defines a cylindrical coordinate system about its surface normal. In this cylindrical coordinate system *radial coordinate* (α)is the perpendicular distance of each point to the normal line, and *elevation coordinate* (β) is the signed distance of the point to the tangent plane to the surface at the vertex. For all the points within the aforementioned support criterion, α and β are computed, and then contribute to the corresponding bins in a 2D accumulator. The closer the *bin size* is to the surface resolution, the matching performance gets better. Based on this statement, we concluded that, the distance of the object to the observer affects the performance, since for most of the depth sensors, the resolution diminishes for farther surfaces.

To test recognition performance of spin images we have followed the same approach as explained in [27]. First, spin images at every point are computed. Next, the average of the spin images is subtracted from each of them. Then, since spin images are highly sparse, Principal Component Analysis (PCA) is applied to reduce the dimensionality of the images. At the test time, the query segment goes through the same process. For each spin image belonging to the query segment, the closest (in Euclidean space) match in the pool of training data's spin images is chosen, if it is closer than 0.8 times the distance to the second match. At the end, geometry checking is performed to prone inconsistent matches. The object that has the most number of matches is reported as candidate match. In our implementation, image widths are 8 and therefore the histograms have 153 bins.

Since spin image computation was fast enough and our database was small, we computed them at every point, for applications that database is big or the point clouds are dense, one might use sampling to reduce computation burden. But it is not clear if keypoints can be used instead of random sampling. It is worth mentioning that spin-images as descriptors have been used in a variety of applications. For example, in [16] it has been used in combination with other shape and contextual features (e.g. estimated height, volume, ...) for object recognition in urban environments.

5.2 Object persistence modelling and data collection

The previous sections describe a method for locating interesting regions and for matching these regions to known named objects that are being tracked within the persistence system. However, there are a number of outcomes from each of these modules, and we combine their outputs with a final top-level reasoning procedure in order to maintain the system's state of the objects over time. When our system enters a new environment, it is unaware of the presence of any objects. It initially scans the world and simply memorizes the current 3D geometry. Object discovery for our system begins when objects begin to move. This triggers the change detection system, and initially each changed object is marked as *new* since the database of known objects is empty. The detection of *new* objects results in the human user being queried for a label: "What is this object?". The user may choose to ignore the object, in which case no model is learned, or a name may be entered: "This is a dragon". Once an initial set of objects is known by the system, further visits to the same scene will continue to result in newly moved objects, and there are now several outcomes possible:

1. The location where an object previously existed is now unoccupied.
2. A previously unoccupied region now contains a known object.
3. A location which previously contained one object now has slightly different geometry sensed, and this matches well to a different object.

4. Any type of change is detected where the new local signature cannot be found in the existing database.

Our top-level reasoning system assigns the following outcomes to the cases above: 1) Indicates that a known object was removed from the space. Notice that, in this case, a part of the scene background that was unobserved before, now will be uncovered. Cases 2) and 3) indicate that one or more known objects have been moved, in which case the object persistence database is updated, and the new location for each object is now utilized for tasks. The feature vector extracted from the new view is merged with the previous appearance model as has been described above. Case 4) indicates that a previously un-modelled object has appeared in the scene. The reasoning system prompts the user for a label and adds this information to the persistence database.

In conclusion, recognition essentially requires a database to match the observed objects to previously labeled objects. Plus, data annotation requires human intervention (assuming the robot does not have access to other resources, such as web.). Moreover, unlike human captured photographs which usually cover all the object in one shot, robot collected data is not guaranteed to have this characteristic. As a result, it is the best that the robots use the data that they have collected, themselves.

Given a database of annotated objects, there are several outcomes from the nearest neighbour matching approaches that we have taken. First, the changed region might unambiguously match to one of the persistent object features that have already been learned by the system. In this case, the recognition system returns a confident detection result for later processing by the top-level persistence component. Another common case is that the changed region matches with a large distance to a number of learned features. We apply a threshold to the nearest neighbour distance, therefore, the recognition module returns the result that the changed region is likely from a semantic category that has previously not been seen by the system. This region can be passed to the user in a query to obtain a new label.

Our object model also allows each object's appearance to be expanded over time. This is an important feature because the system often observes different viewpoints of an object at each time stamp, and the features extracted from each of these viewpoints may not be identical (i.e. in the case of non-symmetric objects). By merging the feature vectors observed over time, the system continues to increase its confidence in the representation of each object, which can provide improvements in accuracy as time progresses. Future work, should focus on building classifiers that learn different classes for the set of uncorrelated viewpoints of the object. Moreover, at the test time, each view of the object votes for one object type, and the collection of votes determine object's class. Further work can focus on, selecting the new views that can clarify ambiguous cases.

Since our system is designed for life long use, and it emulates online learning, we have conducted an experiment in which we input the data collected at observations, each at once. We expect an increase in recognition performance as more observations are added to the database. Although at the same time noisy observations might disturb detection.

As a note for future work please consider that, in case of occlusion, the object's cluster for that particular view is smaller than the object's full model. The object's cluster from an occluded view should not be added to the database. After filtering the occluded views, there are 2 cases for which the objects viewpoint feature histogram should be added to the tree. First, if the viewing angle is more than a threshold distance to the previous key-view (the first view is a key-view, other key-views will be added to the set of views iteratively by following these 2 rules.), in other words, only the segments that are lightly correlated with the segments that are already in the database, should be added. Second, if the histogram distance between the current view and the previous key-view is bigger than a threshold. These rules apply to the views collected from a specific observation. To add views between observations, if the histogram distance, between the new view and the nearest histogram already recorded for the object, is bigger than a threshold, the view should be added to the set of primary views.

5.3 Sensors overview

To acquire 3D position of points in the world various technologies have been developed: Time of Flight (ToF) camera, laser range finder, Stereo camera, Infra Red (IR), and Kinect. Although their working principles differ, they all sense the distance between the sensor and points in the surrounding environment. There are many factors to be considered in order to choose a proper sensor for a specific application. For example, Kinect, which was released to the market recently, is perhaps the cheapest, while the laser scanner has the widest working range. But to acquire a 2D scan, systems equipped with Laser scanner often have to pause and scan the world, in other words sensing on the fly is not possible.

Principally ToF camera senses the range based on the round trip time of light. It provides a depth map with the same frame rate as cameras and no extra-computation is required. Their measurements are usually of lower quality compared to stereo cameras. And they are affected by random and systematic noise.

In this work, we primarily have used Kinect sensor. Note that the algorithms developed in this thesis are not sensor dependent and the point clouds fed into the the algorithm could be generated from any of these sensors. Next sections are brief descriptions of *Stereo camera* and *Kinect*.

5.3.1 Stereo camera

Stereo vision is an imitation, of the approach taken by human vision. In a binocular/stereo camera setting depth can be perceived from the displacements of the points between the images. Therefore the stereo matching quality is highly correlated with the pixels matching module's success.

There are some situations in which window matching between images fail. The two main reasons are *occlusion* and *lack of texture*. Computing depth for regions which are occluded in one image and revealed in the other is not possible without reasoning about the objects that they correspond to. For lack of texture, projecting *artificial* patterns on the scene, proved to be a practical solution. Sometimes, a calibrated (w.r.t. camera) projector is deployed on the robot for this purpose. The characteristics of the patterns have been studied by many groups in past decade.

The patterns are evaluated based on their *self-similarity*. In theory, the more self-similar a pattern is, the number of false matches increases, and vice versa. The proposed patterns are, as simple as, a *random* [40] 2D arrays or as sophisticated as *De Bruijn* [35] pattern. Konolige [31] combines Simulated Annealing with a set of Hamming codes that have a minimum distance with each other. Algorithm starts with a random pattern and searches through the space of possible patterns.

At each step it picks the two blocks (in the pattern) that have minimum Hamming distance and randomly swaps two of their dissimilar pixels. The change is accepted with a probability, and decreases by a exponential factor as the algorithm advances. However the performance mainly depends on the self-similarity of these patterns, other factors are camera and projector's imperfection, as well as, phase difference between them [31].

Even though projected patterns proved to be beneficial for untextured surfaces, their effect is limited to the projection power, and ambient light can diminish their visibility. Stereo camera has been used in both indoor and outdoor applications. Its passive intrinsic (used without projector) is favourable in cases that active sensors, like Laser scanner, cannot be used due to safety issues, *e.g.* medical applications.

Stereo vision is not affected by ambient light as much as Kinect is. Their speed is constrained by rectification and matching algorithms that they use. But currently most of this is done in the camera's hardware internally, which makes it fast. For example the Bumblebee2® camera works at about 30 Hz.

5.3.2 Microsoft Kinect

Kinect was originally designed by PrimeSense® for the game industry, but shortly after its release it was widely used by the robotics community to obtain 3D coloured point clouds. Kinect provides 640×480 coloured and depth images with the rate of approximately 30 frames per second [52]. The depth map reconstruction is performed via proprietary technology from PrimeSense. As described in the Kinect patent by Freedman et al. [15], an astigmatic optical element, that has different focal lengths at various meridional planes, is used to produce a 2D IR pattern. This pattern is projected onto the scene. The spots on the pattern become elongated depending on the distance of the objects to the device. The direction of major axis and the respective length of minor and major axes of the resulting ovals determine the distance to the object.

The practical range of operation is between 0.6 to 6 meters, and its angular field of view is 57° horizontally and 43° vertically [3].

As illustrated in Figure 5.1, the RGB camera and the depth sensors are separated apart by a few centimetres. Therefore, in order to correspond a depth value to

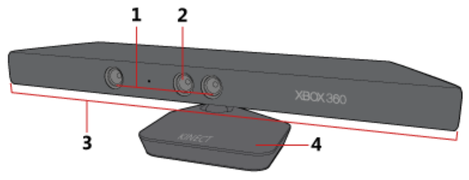


Figure 5.1: 1. 3D depth sensors, 2. RGB camera, 3. multiple microphones, 4. motorized tilt. [1]

each pixel in the RGB image a precise calibration is required. According to Janoch et al. [24] calibration parameters are different for each unit, however some open source calibration packages are available online, e.g. [10].

Chiu et al. [11] experimented with a Kinect sensor and has reported its failure cases in measuring depth: transparency, specularity, dark objects under flat viewing angle, and reflection of projected dot pattern. Moreover, they noted that the depth sensed at the border of small objects is very noisy, because the depth resolution sensed is in order of centimetres. Low depth resolution also leads to missing small and narrow objects.

To clear the point clouds from noise, mainly we have used two types of filters: passthrough filter, and statistical outlier removal (implemented in [47]). A passthrough filter essentially takes the range to be filtered and removes all the points beyond that range. Since the data beyond approximately 3 meters are very noisy and also we focused on close range data, we usually filtered all the range images beyond 3 meters in the direction perpendicular to the device. Statistical outlier removal was also used to remove the noise and small residual segments that are resulted after subtracting the clouds.

Chapter 6

Experiments

A series of experiments are designed to demonstrate the capabilities and limitations of attention and object recognition modules described in previous Chapters (Chapter 4 and Chapter 5).

For our application the testing environment can be as realistic as a kitchen, or it can be synthesized. In either case, the novelty of our approach is to not rely on planar supporting surfaces. There are 3 common points in all the experiments: 1) The objects are placed on top of various supporting surfaces. 2) The number of movable objects in each scene varies. 3) In all the experiments the Kinect was set at 120 *cm* above the ground.[1]

The three following experiments are designed to test the attention operator, object recognition and the performance of the system over time. In Experiment 1 we demonstrate the results of aggregating dynamic objects' bounding boxes to form the regions of interest. We also report the results of static background growing, in order to show that the system is able to extract that only given the observations. Then in Experiments 2 and 3 we assume that taking an observation from the clean scene is possible. Experiment 2 focuses on multi-view object detection and Experiment 3 shows the performance of the system over time.

[1]The camera is usually deployed at two distinct heights on home robots: chair's handle level and human eye level. The former is more appropriate for navigation – localization and obstacle avoidance – and the latter is more suitable for object recognition, exploration and manipulation purposes.

6.1 Experiment 1: attention operator

In this experiment, first we set up an environment to test Region Of Interest (ROI) estimation, and second we set up another environment to examine the process of static background growing. Various scenarios that happen in real world have been considered. The static part of the scene is a typical office desk, and the set of mobile objects are: a banana, an apple, two books, two plant pots, a gift box and a cup. The viewpoint is fixed and movable objects are posed in different angles during observations.

6.1.1 Experiment on region of interest growing

This experiment is divided into two sub-experiments: First, the differences between the consecutive scenes have been estimated by the attention operator, and next the sequence of changes have been accumulated to shape the ROI. To achieve the former many scenarios have been considered:

1. Shifting objects a little so they still occupy some part of their former space. This is challenging in particular for objects that have planar surfaces since subtracting point clouds will remove overlapping areas and remainder point cloud can be challenging to recognize. (A region growing segmentation algorithm could use this remainder point cloud as starting seed.) (see Figure 6.2)

2. Switching objects' position. In this case, the system notices the change and passes the changed clusters to the recognition module, thereby, the fact that these are movable objects and they still exist in the scene will be concluded. The object detector must recognize that the object still exists in the scene. (see Figure 6.3)

3. Adding a new object to the scene. Subtracting the point clouds before and after the change will reveal a portion (if it is occluded) or all of the added object. The object detector must distinguish this as a new object and add it to its database. (see Figure 6.4)

4. Uncovering an occluded area. By uncovering an occluded part of the scene

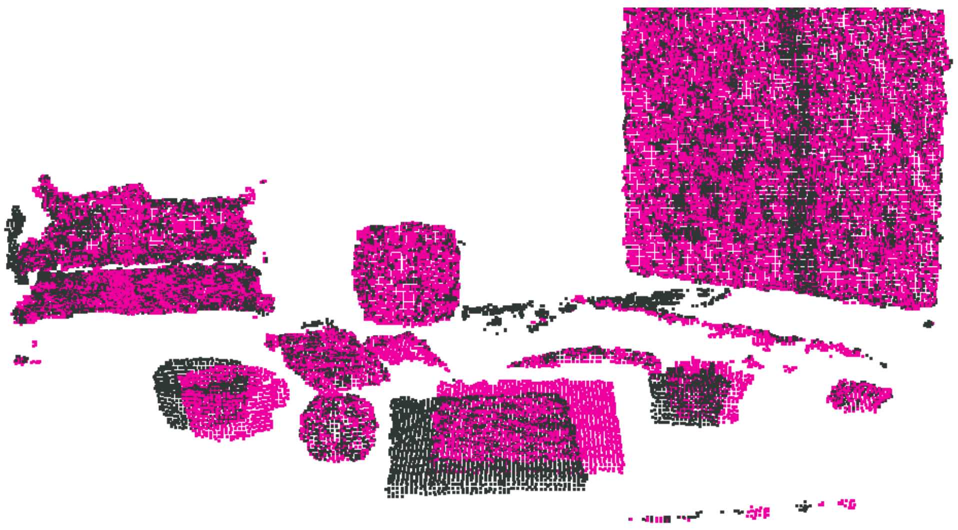


Figure 6.1: The two point clouds, before and after the change, are overlaid in this picture. The purely pink and black parts belong to the scene before and after the change, respectively. The point to notice is that they overlap not only on the static scene but also on the moved objects, specially on the book. In order to illustrate the change, the desk's surface and the wall have been removed from the point cloud.

both static part and possibly dynamic objects might be revealed. To decide whether this space should be added to ROI or not, the system has to rely on future observations. (see Figure 6.5)

5. Replacing an object with a bigger object. This case is very similar to the first scenario depending on the objects' shape. The residual point cloud can cover whole or parts of the new object. (see Figure 6.6)
6. Replacing an object with a smaller object. Depending on the shape of the object the system might fail to detect the new object's shape or extract part of it. This will also uncover a previously occluded area. (see Figure 6.7)
7. Piling objects. This is to show the advantage of our system over object clustering based on plane extraction. (see Figure 6.8)
8. Popping an object from a stack. This has the same objective as the previous

item. (see Figure 6.9)

9. Rotating objects in their place. The cluster extraction task for this scenario is very similar to first item. However recognizing rotated non-symmetric objects can be challenging for object recognition system. (see Figure 6.10)

In the second sub-experiment, the bounding boxes have been aggregated over time to form the region of interest. (see Figure 6.11)

The following pictures are the result of subtracting point clouds, frame by frame. Note that bounding boxes are the projection of 3D bounding boxes to 2D. Red rectangles bound the regions that contain new points and blue rectangles represent the regions containing removed point clusters.

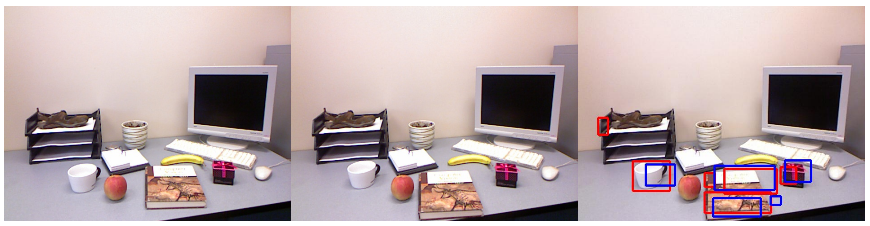
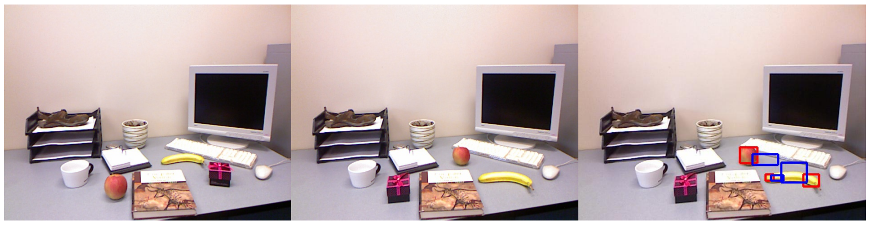

(a) first frame **(b)** second frame **(c)** changes

Figure 6.2: Shifting objects: the book, mug and box have been shifted between frames. The surface of the book, overlaps with itself. Therefore, parts of it have been removed from the scene and parts have been added. On the other hand, since the cup is round, its current surface does not overlap with its former point cloud. The same concept as explained in Figure 4.9

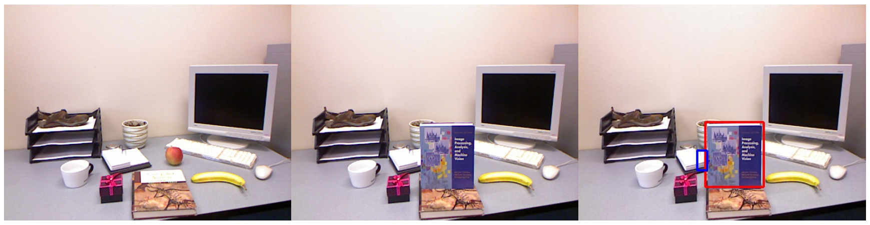
6.1.2 Experiment on accumulating the background

To test the system's performance on background accumulation, we have set up 9 different arrangements of a desk with a number of differently shaped objects. The same process as explained in Chapter 4 has been followed to construct the static background over time. The background is initialized with the first observation and as more observations are inputed to the system, the point that has the farthest distance to the camera at each direction is considered as background. Figure 6.12



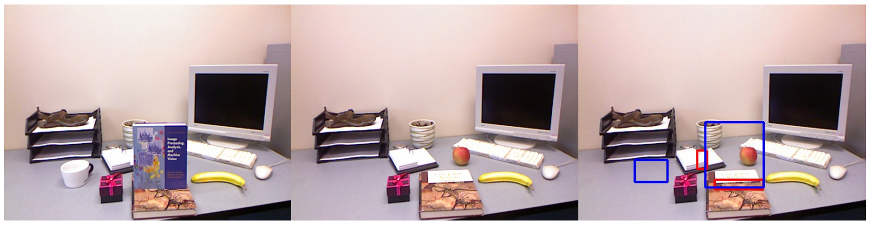
(a) second frame **(b)** third frame **(c)** changes

Figure 6.3: Replacing or switching places: the banana is replaced with the apple, and the box replaced with the banana. Since parts of the banana overlap with the box, it is split into 2 pieces (red boxes).



(a) third frame **(b)** forfth frame **(c)** changes

Figure 6.4: Adding a new object: the book is added to the scene and it occludes the calendar behind it, therefore, a part of calendar seems to be removed.



(a) forfth frame **(b)** fifth frame **(c)** changes

Figure 6.5: Uncovering an occluded area: the book and cup have been removed. They both have been detected and the occluding areas have not been recognized as part of ROI, due to our policy to not assume that the occluded parts are dynamic objects.

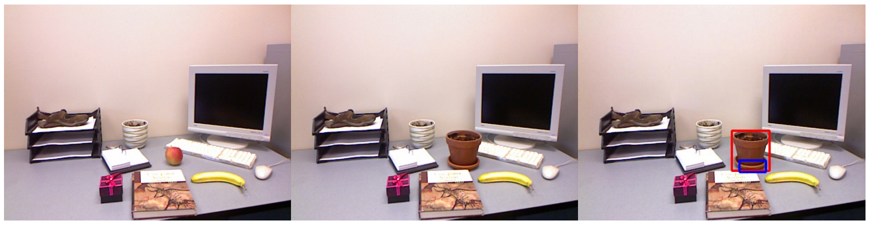

(a) fifth frame **(b)** sixth frame **(c)** changes

Figure 6.6: Replacing an object with a bigger object: the apple is replaced with the plant pot. Their point clouds do not totally overlap, and as a result a part of the apple is considered as removed.

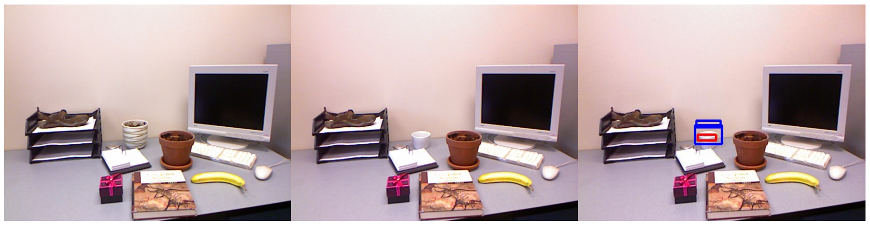
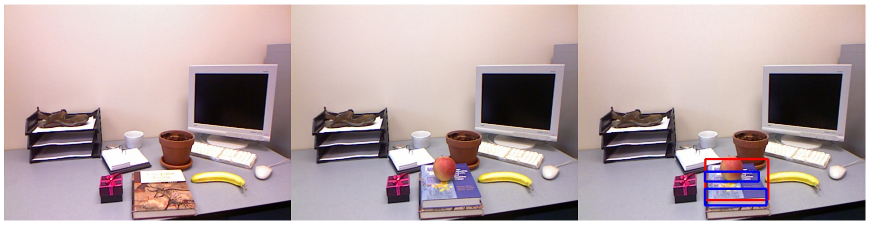

(a) sixth frame **(b)** seventh frame **(c)** changes

Figure 6.7: Replacing an object with a smaller object: this case is very similar to the one illustrated in Figure 6.6. Only the parts that do not overlap are considered as changed regions. Also, the plant pot is segmented out into two pieces, which has resulted in two blue bounding boxes.

depicts the background's evolution over time. Note that the noise present near object edges results from the input Kinect depth data, which is inherently problematic in these situations. In the majority of regions, objects are correctly removed from the background as soon as they are moved.

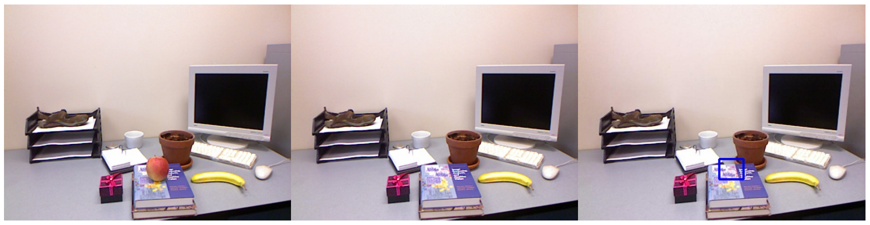
6.2 Experiment 2: object detection

In this experiment, we examined the object detection module in a cluttered office environment. We setup 10 scenes, each contained of a subset of 12 objects, which are depicted in Figure 6.14 (Figure 6.13 represents the projection of the object point clouds to 2D). For each scene, the model is obtained from 5 views. As



(a) seventh frame **(b)** eighth frame **(c)** changes

Figure 6.8: Placing objects on top of each other: a book and an apple are piled over the book. Since the sensor only catches the exterior of the objects which is not occluded, the top of bottom book (blue bounding boxes) is recognized as being removed, although it is still in the scene. It is split into two pieces (blue rectangles) since the operator only compares these two visits, and so it believes that the apple was occluding a part of the book.



(a) eighth frame **(b)** ninth frame **(c)** changes

Figure 6.9: Removing an object from a stack of objects: the apple is removed from the scene and the occluded region behind it is not considered as new.

described in Chapter 4, the detection module constructs a kd-tree [41] from object models[2], and unlike the common practice, in the literature which detects objects in the scene [27, 55], our attention operator provides a point cluster (which believes that it is the whole or part of an object) as query data for detection. Objects have been posed in a variety of orientations to examine the detector's performance in recognizing from different views. To keep the analysis simple and just focus on

[2]Other classifiers (SVM with various kernels, KMeans, and Nearest Neighbours with various distance metrics) have been explored in Rusu [46], for synthetic data with and without added noise. Their results show that after SVM, KNN has the best performance.

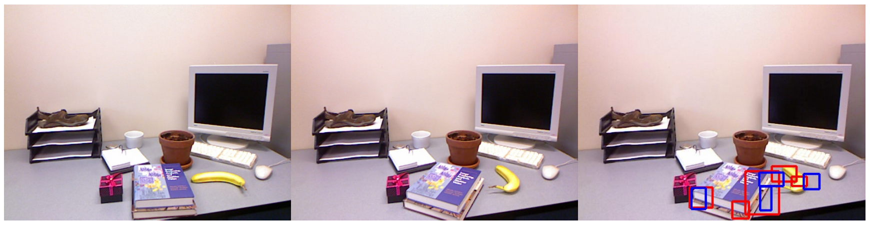

(a) ninth frame **(b)** tenth frame **(c)** changes

Figure 6.10: Shifting and rotating objects: the book and the banana have been rotated, and since they overlap with their previous point clouds, the added and removed regions have been split into a number of small surfaces.

Table 6.1: Number of overall and cluttered instances

	D	CA	S	P1	P2	G	PH	BO	MU	BW	CU	BI
overall	8	9	7	8	6	5	4	2	5	10	8	7
cluttered	0	2	0	3	0	0	2	2	1	7	0	2

object detection performance, each visit has been compared to the clean scene (only the static background). We could not report true-negatives for detection with global descriptors (VFH and VOSCH), because there is always a closest neighbour to each model, and rejecting matches based on histogram distances is not applicable for our small dataset. Perhaps a bigger dataset with a collection of objects that cover a variety of sizes, could be useful to pick a distance threshold.

Table 6.1 contains the number of times each object has appeared in the scenes paired with the number of times that it was occluded. The decision whether an object is occluded or not was determined by visual inspection. A segment marked as occluded if it was covering only a part of the object, and it was hard for a human to identify the segment. Overall, 79 object instances were segmented out of the scenes and fed into a kd-tree for nearest neighbour matching.

Table 6.2 and Table 6.3 represent the closest and second closest matches, respectively. As Table 6.2 illustrates, in most cases the closest match belongs to the correct class. Relatively few mismatches are due to occlusion or in cases where the

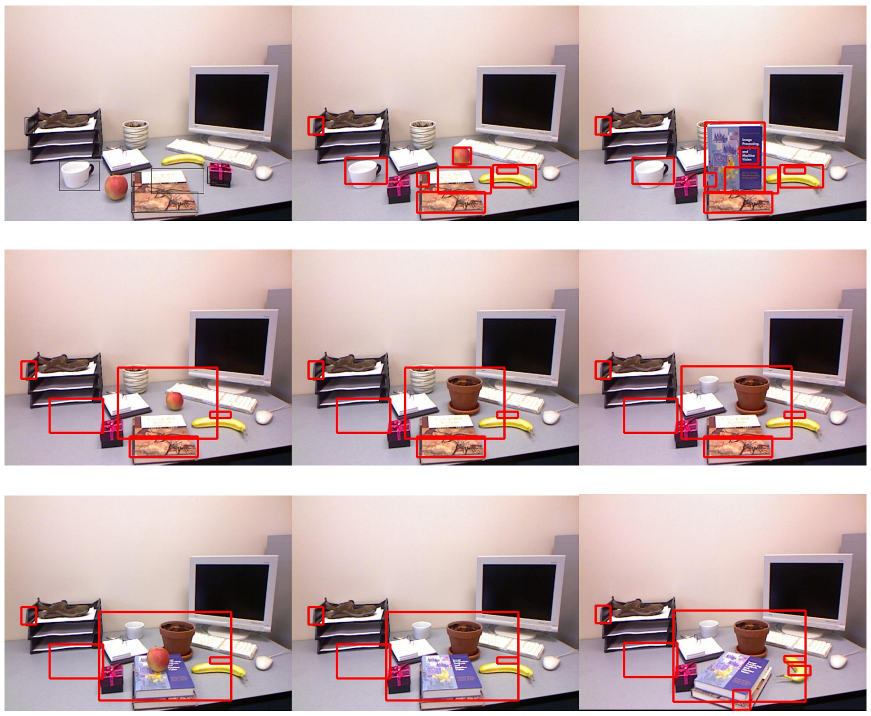


Figure 6.11: ROI growing by attention operator: the sequence of visits starts from the scene on the top-left and ends with the one on the bottom-right. The spaces that contain new point clusters are added to the current ROI at each visit. The final ROI contains most of the working area on the desk. The incorrectly marked region on the paper box has been added to ROI and never removed, this indicates the effect of sensor's failure on the ROI reasoning.

objects had similar shapes.

For example, one of the instances of cat (CA, Figure 6.15a) in the database was only the cat's head, which has a very similar shape to the tea pot (P2, Figure 6.15b) (considering Kinect's low depth resolution). Also, in some cases round objects such as cup, bowl, mug and plant pot are mistakenly matched to each other. Moreover, there was a cat's tail in the database, which matched with the bird or spray in two cases.

The second closest match is not always the correct match. As a result, Table 6.3

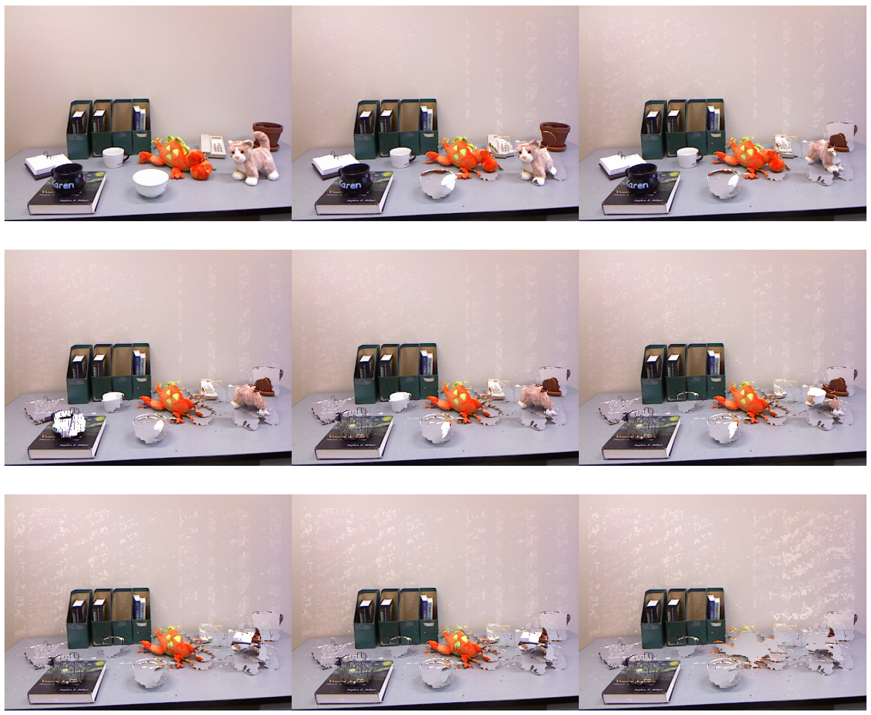


Figure 6.12: The process of updating the 3D model of the background.

looks more scattered. In some cases incorrect second matches are relatively farther compared to the first match. For example, the match between the bowl and bird or cat are relatively farther than the first matches. This said, thresholding the second match does not sound reasonable. The results confirm that, in a very few (only 5) cases (and usually[3] for bowl and bird classes) the second match is correct while the first match is incorrect, therefore it is safe to ignore the second closest neighbours, while using VFH. The last note is that, since the box appeared only in 2 scenes, the second match was always an incorrect match.

The same analysis applies to VOSCH signatures. Tables 6.4 and 6.5 contain closest and second closest matches, accordingly. The number of false matches is very low, as is illustrated in Table 6.4. In the case of VOSCH, only in 2 cases (out of 9 wrong closest matches) the second match is correct. False matches mostly

[3]One other examples is the match between the cat's rear and its front (See Figure 6.17).

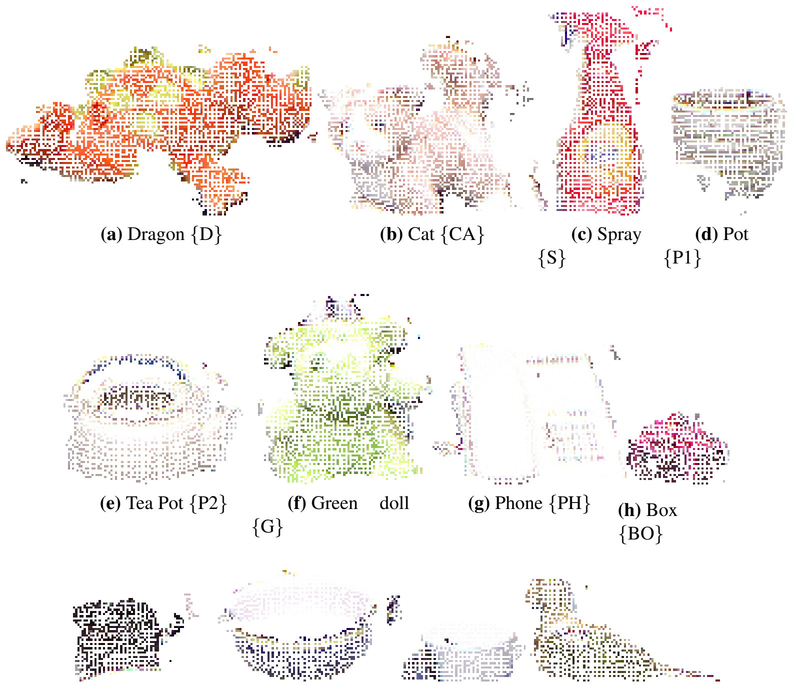
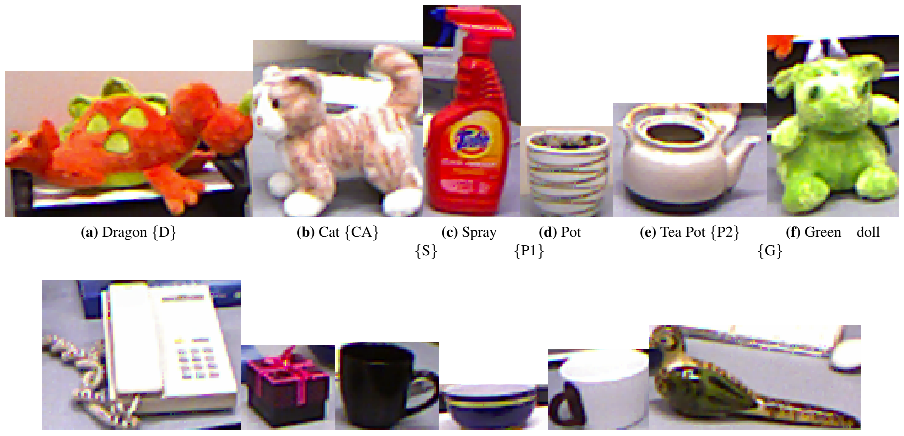

(a) Dragon {D} **(b)** Cat {CA} **(c)** Spray {S} **(d)** Pot {P1} **(e)** Tea Pot {P2} **(f)** Green doll {G} **(g)** Phone {PH} **(h)** Box {BO} **(i)** Mug {MU} **(j)** Bowl {BW} **(k)** Cup {CU} **(l)** Bird {BI}

Figure 6.13: Sample point clouds of the objects used in Experiment 2 and Experiment 3. An imperfect registration results in a noisy point cloud.

happen for occluded objects. A comparison between Table 6.2 and Table 6.4 shows that VOSCH's false matches does not happen between objects with different colours, unless their shapes are very similar. In other words, VOSCH takes advantage of the colour histogram encoded in its signature.

In 5 cases VFH matches were false, but VOSCH matched correctly, whilst with VOSCH 3 matches were wrong and VFH correctly matched. In 6 cases they both made mistakes; these were all occluded instances. Therefore for this data we found that global signatures fail to detect partial views of objects.

We have experimented with spin image as a local feature. The process that is explained in Section 5.1.3 has been followed to find the candidate matches for query objects. According to [26] the point cloud's density influences the spin image's detection performance. Which given the limited angular resolution of the


(a) Dragon {D} (b) Cat {CA} (c) Spray {S} (d) Pot {P1} (e) Tea Pot {P2} (f) Green doll {G}

(g) Phone {PH} (h) Box {BO} (i) Mug {MU} (j) Bowl {BW} (k) Cup {CU} (l) Bird {BI}

Figure 6.14: Automatically extracted image patches of the objects used in Experiment 2 and Experiment 3

Table 6.2: Closest match with VFH

	D	CA	S	P1	P2	G	PH	BO	MU	BW	CU	BI
Dragon	8											
Cat		7	1		1							
Spray	1		6									
Pot		1	1	6								
Tea pot		1			5							
Green doll						5						
Phone						1	3					
Box								2				
Mug				1					4			
Bowl										8	2	
Cup											8	
Bird		1		1								5

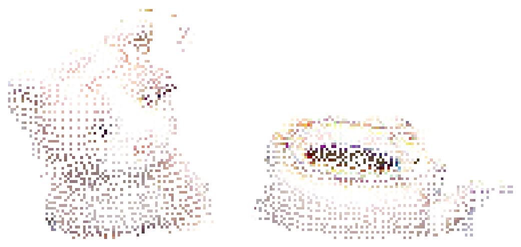
Kinect (compared to a typical laser scanner) the results are poor for spin images.

In all the cases that VOSCH errors, spin images either does not report a candidate match or recognizes the object correctly. On the other hand when VFH fails, spin images might succeed, fail or do not return any matches. Among all classes box and mug never matched to the correct instances.

According to Table 6.6 often small objects are matched to bigger objects. For example, the plant pot is matched to the dragon 4 times, mainly because the dragon

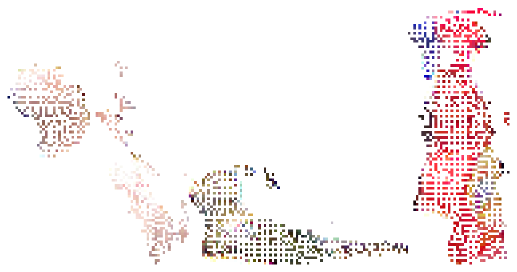
Table 6.3: Second closest match with VFH

	D	CA	S	P1	P2	G	PH	BO	MU	BW	CU	BI
Dragon	7	1										
Cat	1	7			1							
Spray		1	6									
Pot			1	6								1
Tea pot					6							
Green doll						5						
Phone			2			1	1					
Box		2										
Mug				2					2		1	
Bowl		1								6	1	2
Cup				1					1		6	
Bird				2								5



(a) Cat's head **(b)** Tea pot

Figure 6.15: Cat's head matched with tea pot, rather than an un-occluded cat. This is an example of VFH's failure to detect occluded objects.



(a) Cat's tail **(b)** Bird **(c)** Spray

Figure 6.16: Cat's tail matched with bird and spray, rather than an un-occluded cat. This is an example of VFH's failure to detect occluded objects.

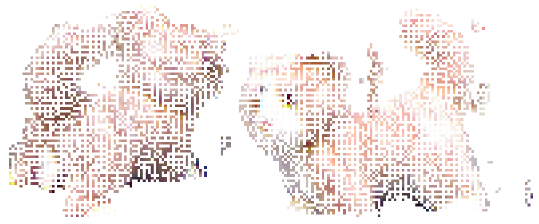

(a) Cat's rear view **(b)** Cat's front view

Figure 6.17: Cat's rear view matched with its front. It is probably due to its symmetry.

Table 6.4: Closest match with VOSCH

	D	CA	S	P1	P2	G	PH	BO	MU	BW	CU	BI
Dragon	8											
Cat		7		1							1	
Spray			7									
Pot				6			1				1	
Tea pot					6							
Green doll						5						
Phone				1			2				1	
Box								2				
Mug									5			
Bowl										7	2	1
Cup											8	
Bird												7

has a rounded shape in many places. This phenomenon is due the fact that bigger shapes are composed of smaller shapes that might be similar to a small object. On the other hand, the big shapes might be partly occluded in some observations which results in a point cloud that is smaller than the model in the database. To resolve the first issue, we can bound the size ratio between the query object segment and the models in the database, but this can eliminate correct matches to occluded object segments. To investigate this tradeoff, we thresholded the size ratio between the candidate matches. Figure 6.18 presents the false-negative and true-positive ratios with respect to allowed minimum scale ratio. There is a trade off between the low false-negative and high true-positive. 0.7 is the ratio that has the highest true-positives and a relatively low false-negative. Thresholding the candidate matches

Table 6.5: Second closest match with VOSCH

	D	CA	S	P1	P2	G	PH	BO	MU	BW	CU	BI
Dragon	7	1										
Cat		7		1	1							
Spray			7									
Pot		1		5	1							
Tea pot					6							
Green doll						5						
Phone				1	1		1				1	
Box												2
Mug									5			
Bowl								2		3	4	1
Cup											8	
Bird										1		6

Table 6.6: Closest match with spin images

	D	CA	S	P1	P2	G	PH	BO	MU	BW	CU	BI	WM
Dragon	8												
Cat	1	6				1							1
Spray	1	1	4				1						
Pot	4		1	2					1				
Tea pot	1	1			4								
Green doll	2	1				2							
Phone			1				3						1
Box													2
Mug	1			1	1								2
Bowl		1	1							4			5
Cup	1										4	1	3
Bird	1	1			1								4

based on this parameter means that if more than 30 percent of an object is occluded, the system will not recognize it.

Table 6.7 presents the matches after thresholding candidate matches with the scale ratio bellow 0.7. As we expected false matches have decreased, but the false-negative ratio has decreased.

6.3 Experiment 3: detection performance over time

In this experiment, the performance of object detection module for a growing database has been examined. The sequence of visits are the same as Experiment

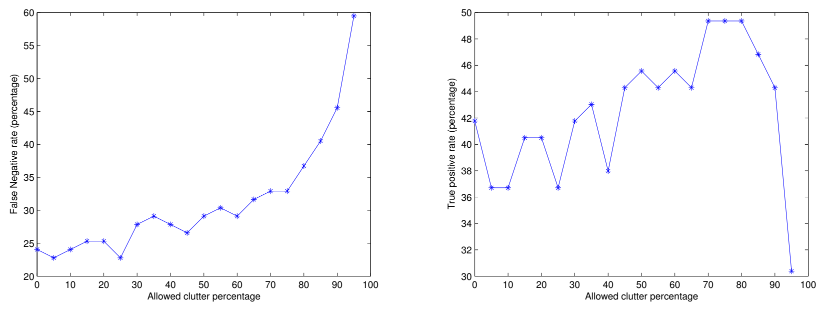


Figure 6.18: False negative and true positive graphs with respect to the allowed minimum ratio between the sizes of matched objects. These graphs show that if we assume up to 30 percent of the objects are allowed to be occluded, the false negative will be 33 percent and true positive is 49 percent.

Table 6.7: Detection results for spin images, minimum size ratio allowed 0.7 (true positives: 51% false negatives: 34%)

	D	CA	S	P1	P2	G	PH	BO	MU	BW	CU	BI	WM
Dragon	7	1											
Cat	4	2				1							2
Spray			6				1						
Pot			1	6									1
Tea pot			1		4								1
Green doll						5							
Phone			1				2						1
Box													2
Mug			1								1		3
Bowl										4			6
Cup											3		5
Bird												1	6

2. The database was preoccupied with the object point clouds from the first visit, and starting from the second visit, sequentially, first the object point clouds have been matched to the objects that were already in the database, and then were added (with their correct labels – assuming that a human will correct the false matches) to the database.

Tables 6.8, 6.9 and 6.10 represent the detection results for all (total number of 10) visits. A comparison between the number of false matches in Table 6.2 and

Table 6.8: Over time recognition performance with VFH

	Day$_1$	*Day*$_2$	*Day*$_3$	*Day*$_4$	*Day*$_5$	*Day*$_6$	*Day*$_7$	*Day*$_8$	*Day*$_9$
Dragon	🟩	🟩	🟥 CA	🟩	🟩			🟩	🟩
Cat	🟩	🟩	🟥 D	🟩	🟩	🟥 P2,S			🟩
Spray	🟥 PH	🟩	🟩		🟩	🟩			🟩
Pot	🟩	🟥 S			🟩	🟩	🟥 CA	🟩	🟩
Tea pot	🟥 P	🟩	🟩	🟩			🟩		
Green doll					🟥 CA	🟩	🟩	🟩	🟩
Phone	🟩	🟩							
Box								🟥 CA	🟩
Mug	🟩		🟩	🟥 P1					🟥 P1
Bowl	🟩	🟩		🟥 CU	🟩	🟩 BW,BI	🟩	🟩	
Cup	🟩	🟩	🟩	🟩			🟥 MU	🟩	🟩
Bird	🟩	🟩		🟩		🟥 CA	🟥 P1	🟩	

Table 6.9: Over time recognition performance with VOSCH

	Day$_1$	*Day*$_2$	*Day*$_3$	*Day*$_4$	*Day*$_5$	*Day*$_6$	*Day*$_7$	*Day*$_8$	*Day*$_9$
Dragon	🟩	🟩	🟩	🟩	🟩			🟩	🟩
Cat	🟩	🟩	🟩	🟩	🟩	🟥 P2,P			🟩
Spray	🟩	🟩	🟩		🟩	🟩			🟩
Pot	🟩	🟥 CU			🟩	🟩	🟥 PH	🟩	🟩
Tea pot	🟩	🟩	🟩	🟩			🟩		
Green doll					🟥 CA	🟩	🟩	🟩	🟩
Phone	🟩	🟩							
Box								🟥 BI	🟩
Mug	🟩		🟩	🟩					🟩
Bowl	🟥 BI	🟩		🟥 CU	🟩	🟥 CU,BI	🟩	🟩	
Cup	🟩	🟩	🟥 PH	🟩			🟩	🟩	🟩
Bird	🟩	🟩		🟩		🟩	🟩	🟩	

Table 6.8 indicates that VFH improves as more instances are added to the database. On the other hand Table 6.9 shows no difference in the number of false matches. The true-positive ratio for spin images increased from 50.6% in Table 6.7 to 60% in Table 6.10. Other than two instances of bowl class, spin images do not show any advantage over VFH or VOSCH.

6.4 Conclusions

With this set of experiments we have tested the attention operator, static scene modelling, object recognition and performance of the object detection process over

Table 6.10: Over time recognition performance with spin images

	Day$_1$	*Day*$_2$	*Day*$_3$	*Day*$_4$	*Day*$_5$	*Day*$_6$	*Day*$_7$	*Day*$_8$	*Day*$_9$
Dragon	🟩	🟩	🟩	🟩	🟩			🟩	🟥
Cat	🟩	🟩	🟩	🟩	🟩	🟥			🟩
Spray	🟥	🟩	🟩		🟩	🟩			🟩
Pot	🟩	🟥			🟩	🟩	🟥	🟥	🟥
Tea pot	🟩	🟩	🟥	🟩			🟩		
Green doll					🟥	🟥	🟩	🟩	🟩
Phone	🟥	🟥							
Box								🟥	🟥
Mug	🟩		🟥	🟩					🟩
Bowl	🟥	🟩		🟩	🟩	🟥	🟥	🟩	
Cup	🟩	🟩	🟥	🟩			🟥	🟥	🟩
Bird	🟥	🟩		🟥		🟥	🟥	🟥	

time. For the three feature detectors that we have examined, overall VFH, VOSCH and spin images correctly identified 85, 89 and 47 percent of the objects, respectively. We speculate that the relatively low performance of spin images is due the low resolution of the Kinect. The VOSCH's success in outperforming the other two feature detectors is partially due its use of colour. In addition it is designed to be robust to occlusion. Given the fact that in some settings previously observed objects were posed from a totally new angle, the results are very promising. Also, the performance of the object recognition does not seem to be affected as the database grows, however assessing the usefulness of adding every new segment to the database requires more trials with a bigger collection of objects, which we leave for future investigation.

Chapter 7

Conclusion and future directions

This thesis presented an interactive system for lifelong object understanding without the requirement of prior data collection and training. We achieved this by focusing the attention of a system on the dynamic foreground of a scene while ignoring static background regions using 3D data such as the type available from an RGB-D sensor. Interesting regions are described by an interactive object modelling component, which assembles a multi-view appearance model for each of the items that a user is interested in, on-the-fly, which replaces the pre-training required by other methods.

Our complete system addresses the technical challenges of registration, change detection, region analysis, and multi-view specific object matching. Results show that for tabletop scenes, our approach is able to effectively separate object regions from background, and that objects can be successfully modelled and recognized during system operation. Limitations and strengths of each component have been assessed with experiments.

There were a number of directions that given the time constraint we did not investigate and are left as future work:

- Building background model from multiple views is more challenging than observing the scene from a fixed view point. Our assumption about the static background breaks down if the scene is observed from multiple views – the static parts are not necessarily the farthest points observed at each ray direction anymore. We suggest modelling the components of the 3D background, such as desks, shelves, tables, walls and doors, separately, and integrating those into a single static background model.

- With our implementation, the transformations were noisy. We took a dead-reckoning approach to registration and did not optimize the camera trajectory given the constraints that the overlapping areas in the sequence provided. The most well known approach used for this purpose in the literature is *bundle adjustment*. We expect to get better registration results if such adjustment is performed on the transformations. For state-of-the-art vision based mapping refer to [32].

- The object recognition procedure proposed in this thesis reports a candidate match with the nearest neighbour approach. If a bigger database is collected (for example by running the system in a bigger environment) performing object classification with other techniques could be beneficial. Also an "unable to recognize" flag could be raised in places where feature detectors do not agree on a single label for the query object.

- The size of objects that moved between our observations was relatively small compared to the objects that move in a household or home care facility environment (*e.g.* wheelchairs). It is beneficial to have a model of these objects in order to understand the environment. The primary limiting factor for our prototype was the Kinect's limited working range.

- The logic of growing a region of interest that was proposed in this thesis can be beneficial for systems that need to attend to regions that dynamic objects are or can be placed on. For example a manipulator needs to know the places that an object can be placed on, or an object search system needs to attend to regions that may contain the object that it is looking for.

- Integrating our system with a robust Simultaneous Localization and Mapping (SLAM) system and running the the pipeline in a bigger environment would provide the ultimate functionality that we expect for our system.

Bibliography

[1] Kinect-sensor-components-xbox.com. Website, December 2011. URL http://support.xbox.com/en-US/kinect/more-topics/kinect-sensor-components. → pages 44

[2] Google 3D warehouse. Website, January 2012. URL http://sketchup.google.com/warehouse/. → pages 12

[3] Website, February 2012. URL http://en.wikipedia.org/wiki/Kinect. → pages 43

[4] Robotic 3D scan repository. Website, January 2012. URL http://kos.informatik.uni-osnabrueck.de/3Dscans/. → pages 12

[5] J. F. Allen. *Maintaining knowledge about temporal intervals*, pages 361–372. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. → pages 27, 28

[6] A. Aydemir, A. Pronobis, K. Sjöö, M. Göbelbecker, and P. Jensfelt. Object search guided by semantic spatial knowledge. In *The RSS'11 Workshop on Grounding Human-Robot Dialog for Spatial Tasks*, Los Angeles, CA, USA, July 2011. → pages 8

[7] M. Beetz, Z. C. Marton, D. Pangercic, R. B. Rusu, and A. Holzbach. Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, 2010 2010. → pages 37

[8] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence - Special issue on interpretation of 3-D scenes—part II*, 14:239–256, February 1992. → pages 21

[9] N. Beuter, A. Swadzba, F. Kummert, and S. Wachsmuth. Using articulated scene models for dynamic 3D scene analysis in vista spaces. *3D Research*, 1 (3):20:1–20:13, Sept. 2010. → pages 9

[10] N. Burrus. Kinect RGBDemo v0.6.1. Website, December 2011. URL http://nicolas.burrus.name/index.php/Research/KinectRgbDemoV6?from=Research.KinectRgbDemoV2. → pages 44

[11] W. Chiu, U. Blanke, and M. Fritz. Improving the Kincept by cross-modal stereo. In *22nd British Machine Vision Conference (BMVC)*, Dundee, UK, 2011. → pages 44

[12] H. B. Darbandi, M. R. Ito, and J. Little. Surface signature-based method for modeling and recognizing free-form objects. In *Proceedings of the 3rd international conference on Advances in visual computing - Volume Part II*, ISVC'07, pages 447–458, Berlin, Heidelberg, 2007. Springer-Verlag. → pages 34

[13] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard. Unsupervised discovery of object classes from range data using Latent Dirichlet Alocation. In *Robotics: Science and Systems*, 2009. → pages 34

[14] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision: issues, problems, principles, and paradigms*, pages 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. → pages 21

[15] B. Freedman, A. Shpunt, and Y. Arieli. Distance-varying illumination and imaging techniques for depth mapping, November 2010. → pages 43

[16] A. Golovinskiy, V. Kim, and T. Funkhouser. Shape-based recognition of 3D point clouds in urban environments. In *IEEE 12th International Conference on Computer Vision, 2009*, pages 2154–2161, 2010. → pages 39

[17] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3D scene geometry to human workspace. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1961–1968, May 2011. → pages 12

[18] G. D. Hager and B. Wegbreit. Scene parsing using a prior world model. *International Journal of Robotics Research*, 30(12):1477–1507, Oct. 2011. → pages 8, 9

[19] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988. → pages 22

[20] N. Hawes, M. Hanheide, J. Hargreaves, B. Page, H. Zender, and P. Jensfelt. Home alone: Autonomous extension and correction of spatial representations. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011. → pages 10

[21] E. Herbst, P. Henry, X. Ren, and D. Fox. Toward object discovery and modeling via 3-d scene comparison. In *International Conference on Robotics and Automation*, pages 2623–2629, 2011. → pages 9

[22] E. Herbst, X. Ren, and D. Fox. RGB-D object discovery via multi-scene analysis. In *Proceedings of the 2011 IEEE International Conference on Intelligent Robots and Systems*, 2011. → pages 9

[23] J. Hoey, P. Poupart, A. von Bertoldi, T. Craig, C. Boutilier, and A. Mihailidis. Automated handwashing assistance for persons with dementia using video and a partially observable Markov Decision Procs. *Computer Vision and Image Understanding*, 114(5):503 – 519, 2010. → pages 1

[24] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-D object dataset: Putting the Kincept to work. In *ICCV Workshop on Consumer Depth Cameras for Computer Vision*, 2011. → pages 12, 44

[25] Z. Jia, A. Saxena, and T. Chen. Robotic object detection: Learning to improve the classifiers using sparse graphs for path planning. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2072–2078. IJCAI/AAAI, 2011. → pages 11

[26] A. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997. → pages 55

[27] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:433–449, May 1999. → pages 34, 38, 51

[28] A. Kanezaki, Z.-C. Marton, D. Pangercic, T. Harada, Y. Kuniyoshi, and M. Beetz. Voxelized shape and color histograms for RGB-D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),*

Workshop on Active Semantic Perception and Object Search in the Real World, 2011. → pages 34, 37

[29] A. Kanezaki, T. Suzuki, T. Harada, and Y. Kuniyoshi. Fast object detection for robots in a cluttered indoor environment using integral 3D feature table. In *IEEE International Conference on Robotics and Automation*, pages 4026 –4033, May 2011. → pages 37

[30] T. Kollar and N. Roy. Utilizing object-object and object-scene context when planning to find things. In *IEEE International Conference on Robotics and Automation*, pages 2168 –2173, May 2009. → pages 8

[31] K. Konolige. Projected texture stereo. In *IEEE International Conference on Robotics and Automation*, pages 148 –155, 2010. → pages 42, 43

[32] K. Konolige and J. Bowman. Towards lifelong visual maps. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1156 – 1163, 2009. → pages 10, 13, 64

[33] K. Lai and D. Fox. Object recognition in 3D point clouds using web data and domain adaptation. *International Journal of Robotics Research*, 29: 1019–1037, July 2010. → pages 12

[34] K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *IEEE International Conference on Robotics and Automation*, 2011. → pages 12

[35] J. Lim. Optimized projection pattern supplementing stereo systems. In *Proceedings of IEEE international conference on Robotics and Automation*, ICRA'09, pages 3832–3838, Piscataway, NJ, USA, 2009. IEEE Press. → pages 42

[36] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, ICCV '99, Washington, DC, USA, 1999. → pages 22

[37] Z.-C. Marton, D. Pangercic, N. Blodow, and M. Beetz. Combined 2D-3D categorization and classification for multimodal perception systems. *International Journal of Robotics Research*, 30(11):1378–1402, September 2011. → pages 8

[38] D. Meger, P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, D. G. Lowe, and B. Dow. Curious George: An

attentive semantic robot. In *IROS 2007 Workshop: From sensors to human spatial concepts*, San Diego, CA, USA, November 2007. → pages 11

[39] D. Meger, P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, D. G. Lowe, and B. Dow. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems Journal Special Issue on From Sensors to Human Spatial Concepts*, 56(6):503–511, 2008. → pages 10

[40] T. Molinier, D. Fofi, J. Salvi, Y. Fougerolle, and P. Gorria. Projector view synthesis and virtual texturing. In *International Topical Meeting on Optical Sensing and Artificial Vision*, 2008. → pages 42

[41] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009. → pages 36, 51

[42] J. Ponce, T. L. Berg, M. Everingham, D. Forsyth, M. Hebert, S. Lazebnik, M. Marszałek, C. Schmid, C. Russell, A. Torralba, C. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. In *Towards Category-Level Object Recognition*, pages 29–48. Springer, 2006. → pages 10

[43] R. Rusu, N. Blodow, Z. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384 –3391, September 2008. → pages 35

[44] R. Rusu, Z. Marton, N. Blodow, and M. Beetz. Learning informative point classes for the acquisition of object model maps. In *International Conference on Control, Automation, Robotics and Vision*, pages 643 –650, December 2008. → pages 34, 35, 36

[45] R. Rusu, Z. Marton, N. Blodow, M. Dolha, and M. Beetz. Functional object mapping of kitchen environments. In *International Conference on Intelligent Robots and Systems*, pages 3525 –3532, September 2008. → pages 8, 25

[46] R. B. Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muechen, Germany, October 2009. → pages 25, 51

[47] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011. → pages 22, 34, 44

[48] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 1848–1853, Piscataway, NJ, USA, 2009. → pages 34, 36

[49] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3D recognition and pose using the viewpoint feature histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010. → pages 34, 35, 36

[50] U. Rutishauser, D. Walther, C. Koch, and P. Perona. Is bottom-up attention useful for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 37–44, 2004. → pages 8

[51] B. Sapp, A. Saxena, and A. Y. Ng. A fast data collection and augmentation procedure for object recognition. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, pages 1402–1408. AAAI Press, 2008. → pages 11

[52] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297 –1304, June 2011. → pages 43

[53] T. Southey and J. J. Little. Object discovery through motion, appearance and shape. In *AAAI Workshop on Cognitive Robotics*, 2006. → pages 9

[54] C. Stachniss. *Robotic Mapping and Exploration*. Springer Tracts in Advanced Robotics. Springer, 2009. → pages 19

[55] B. Steder, G. Grisetti, M. Van Loock, and W. Burgard. Robust on-line model-based object detection from range images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 2009. → pages 22, 51

[56] B. Steder, G. Grisetti, and W. Burgard. Robust place recognition for 3D range data based on point features. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. → pages 22, 23

[57] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011. → pages 12, 23

[58] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Point feature extraction on 3D range scans taking into account object boundaries. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011. → pages 22, 23

[59] P. Viswanathan, J. J. Little, A. K. Mackworth, and A. Mihailidis. Navigation and obstacle avoidance help (NOAH) for older adults with cognitive impairment: a pilot study. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '11, pages 43–50, New York, NY, USA, 2011. → pages 1

[60] D. Walther, U. Rutishauser, C. Koch, and P. Perona. On the usefulness of attention for object recognition. In *Workshop on Attention and Performance in Computational Vision at ECCV*, pages 96–103, 2004. → pages 8

[61] W. Wohlkinger and M. Vincze. Shape-based depth image to 3D model matching and classification with inter-view similarity. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4865 –4870, September 2011. → pages 12

[62] H. Zender, Óscar Martínez Mozos, P. Jensfelt, G.-J. M. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, June 2008. → pages 10