Master Thesis
Towards Bootstrapping Robotic Perception of Indoor Environments

Aswin Chandarr A.B
Student Number : 4119541

Supervisors
Prof. dr. ir. Pieter Jonker
Maja Rudinac
Delft Biorobotics Laboratory
Department of BioMechanical Engineering
TU Delft

August 26, 2012

Abstract

Robots have been popular amongst us as Science fiction characters for a few decades now. The inability of the robots to robustly perceive and respond to the real world has been confining them to the laboratories for a long time. This can be attributed to the dynamic nature of the everyday environments where the prelearnt knowledge alone is not sufficient.

The robots can be developed to work autonomously in these situations when they can obtain and update the knowledge of their surrounding on their own without external intervention. This process is termed as bootstrapping and it involves perceiving various aspects of the environment like faces, objects, sound, etc. This is a very elaborate task given the current level of sophistication. It is important for an intelligent robot to be able to comprehend unknown objects in the scene and this thesis focuses on handling unknown objects.

A segmentation based on the 3 dimensional sensory data of the visual scene is implemented. The large planar structures in the scene are identified and the rest of the data is clustered to locate the probable objects in the scene. The appearance of the objects are chromatically and spatially described and are the basis on which they are recognized from the pre-learnt object models.

A generic grasping technique based on the visual tracking feedback has been proposed and implemented to grasp any different object from various common locations.

ACKNOWLEDGEMENTS

The satisfaction and euphoria that accompany the successful completion of any task would be, but incomplete without the mentioning of the people who made it possible.

I would like to thank **Prof. dr. ir. Pieter Jonker** for providing me with the opportunity to pursue my Master thesis with Delft Biorobotics Laboratory.

I extend my sincere and heartfelt gratitude to **Maja Rudinac** for guiding me throughout the course of my thesis. The constant discussions and feedback helped a lot to improve my understanding of the real challenges in development of personal robots.

I deeply thank and honour my colleagues in the DelftRobotics team of Machiel Bruinink, Floris Gaisser, Mukundha Bharatheesha, Mathieu Seban, Susana Pons Rueda and Hans Gaisser for their assistance with software and without whom the thesis would not be complete.

I also should thank Guus Liqui Lung, Wouter Caarls, Boris Lenseigne and Xin Wang of the Delft Biorobotics Laboratory for their critical advices.

I should thank my friends Revathi, Sergio, Nishant for their moral support and enthusiasm during the tough phases of this thesis.

Last but not the least, I thank my parents who have been a pillar of support and providing me courage throughout the course of my academic life.

Contents

List of Figures

Chapter 1

Introduction

It is quite fascinating for any person to look at robots in action. It becomes more enthralling as the spectrum of the functionalities gets elaborate. The process of building a reliable and intelligent robot is engrossing, extensive and almost perpetual. But the satisfaction of taking a small step in this direction has been inspiring generations of robotic researchers.

Though robots are widely prevalent only in the recent decades, the concept of robots has been existing in various forms across different cultures for very long. The following section briefly traces the history of robots.

1.1 History of Robots

Technologically the robots evolved from the water/wind mills which constituted the first automatic machines. This transformed into steam engines during the industrial revolution and into electrical machines in the second industrial revolution. The development accelerated with the invention of computers in the digital era and this forms the backbone of the present day robots. Though sophisticated robots have been built only in the recent years, the idea of robots has been existing since historic times.

Aristotle postulated in his book *Politics*[1] about bringing human equality through *Automatons* someday [1]. He also speculated abolition of slavery made possible as a consequence.

> *There is only one condition in which we can imagine managers not needing subordinates, and masters not needing slaves. This condition would be that each instrument could do its own work, at the word of command or by intelligent anticipation, like the statues of Daedalus or the tripods made by Hephaestus.*

But in practise, the first automatic self regulatory device *clepsydra*, invented by Ctesibius [27] in ancient Greece dates back to around 250 BC. It was an improved water clock with proper feedback and control mechanisms and it was the most accurate clock until Christian Huygens invented the pendulum clock in 17th century AD.

While in 9th Century Su Song devised an automatic water powered astronomical clock [7] in China. It featured a armillary sphere powered by a mechanical clock drive.

A functional mechanical robot that could play music can be dated to the 12th century. Al-Jaziri an Arab inventor developed a hydraulically controlled musical band that could be programmed to play different tunes [23]. A modern replica of these musicians as in Fig.1.1 is perhaps the oldest known programmable automaton.
The first *Humanoid automaton* was designed by Leonardo da Vinci in the 15th century. He designed a robot Knight which could sit, stand, raise its visor and maneuver its arms [10]. This system was operated by a series of pulleys and cables. A reconstruction of this design and its mechanism is shown in the Fig.1.2.

Many small automatons were built in the following centuries. With the advent of industrialization, Robots were developed to automate the manufacturing processes. After the WW2,

[1] (ca. 322 BC, book 1, part 4)

Figure 1.1: Earliest Programmable Automatons

Figure 1.2: Da Vinci's Robot Knight Model

as the electronics started to become sophisticated, robots were widely deployed in industries handling massive assembly lines. With their increased usage in factories, robots/automatons slowly started becoming prevalent among masses through plays and movies.

The word "robot" has its origin in Czech word "robota" where it means "Servitude". It was formally used for the first time by writer Karel Capek in his play *Rossuum's Universal Robots*. They were conceived as automatic machines that would assist humans. The definition and the limitation of functionalities of the robot have been evolving with the technological progress.

This initiated the idea of household service robots in science-fictions. It became popular with *Asimov's* famous work of *I'Robot*. More movies like *StarWars* series have portrayed robots helping humans in everyday life in the near future, Fig.1.3. Though robots have been expected to be in common households for quite long now, they are still expensive and are confined to the laboratories. But, when the robots can be brought out of the laboratories, there exist myriad applications to assist and improve human life.

1.2 Scope of Robotics

The application of robots is quite ubiquitous. The development is currently in a nascent stage like the computers were two decades earlier. But in a decade from now, robots will be pervasive in every sphere of human life with the limits of utility bounded only by the creativity of the developers. This will be similar to the way computers have become universal now. They can assist in everyday tasks by co-existing with humans and also they can operate in non-human friendly conditions as well. This will expand the realm of human dexterity and can also lead to previously unknown territories. Quite a lot of robots have already been deployed in commonplace and specialized applications.
Some of the current applications include

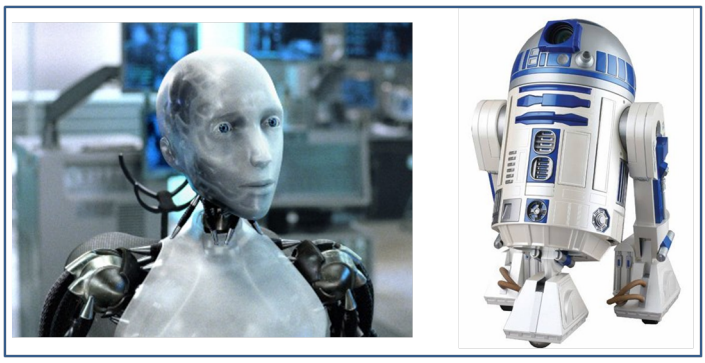

Figure 1.3: Science Fiction Robots

- Extra Terrestrial Exploration robots
- Under Ground and Deep sea probes
- Industrial assembly robots
- Surgical assistance robots
- Amusement Toys, etc.

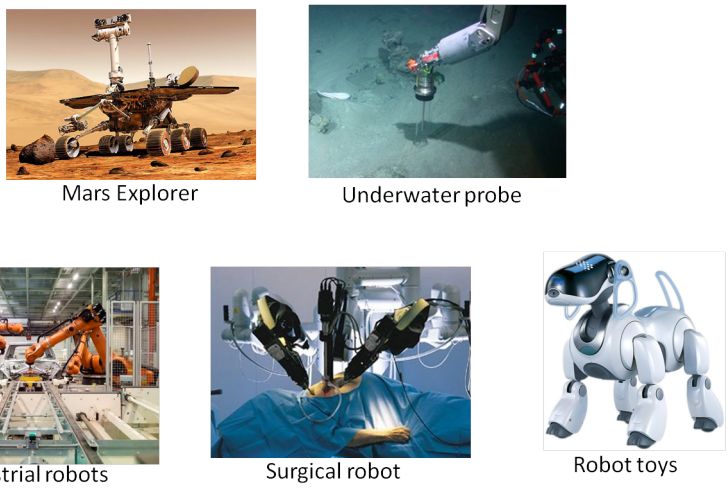

Figure 1.4: Existing robotic applications

Apart from a lot of scope for improvement in the existing robots, there also exists a plethora of new applications like *personal robots* where further *intelligence* and *robustness* is required. An integrated development of hardware and software will eventually lead to this stage.

The electronics and mechanics which constitute the hardware are technologically advanced to be deployed in practise. The Software part which controls these has not yet reached that sophistication. In the recent years with the increasing popularity of the *Open Source* software like ROS[2], PCL[3], etc. the pace of robotics research has been accelerated rapidly. This is mainly due to the unified development of common platform on which new algorithms can be integrated rather than reinventing the wheel every-time.

Hence, it is the right time to learn from and contribute to the vast global pool of knowl-

(a) ROS (b) PCL

[2]Robot Operating System: http: //www.ros.org
[3]Point Cloud Library: http://www.pointclouds.org

edge. This will foster bringing *personal service robots* from Science-Fiction to reality. The essence of a thesis lies in understanding the broader perspective and deriving an clearly defined smaller objective problem statement leading towards it. This starts with a scientific comprehension of the current state of technology and identifying the gap to be bridged. The following section summarizes this aspect.

1.3 Action and Perception

Technically the domain of Robotics research can be segregated into two spaces as

1. Perception Space
2. Action Space

Perception Space

Quite apparent from the name, this domain deals with understanding the environment by interpreting the sensory information. There can be wide range of devices to sense different environmental stimuli like auditory, visual, tactile information. The hardware part of the perception includes devices like

- Camera
- Microphone
- Tactile and Proximity Sensors
- Laser Scanners, etc.

These devices can only convert the external stimuli into digital data. The *perception space* also involves interpreting the environment by manipulating this data. This is the software part which involves aspects like

- Image Processing
  - Face Recognition
  - Object Recognition
  - Gesture Perception, etc.
- Natural Language Processing
- Obstacle identification, etc.

While these facilitate comprehending the environment the robot also has to respond differently to diverse forms of interpreted information. This domain of robotics is the *action space*.

Action Space

The robot also has to act in and on the environment to be utilitarian. This constitutes the *action space* where the robot is animate using different actuators like

- Grippers
- Arms
- Wheels, etc.

These can be seen as in Fig.1.3.
This also involves quite elaborate software behind any motion. There are several modules like

- Navigation
- Arm motion planning
- Grasp planning
- Speech synthesis, etc.

A tight linkage between these two wide domains is vital and is explained in the following section.

(c) Sensors

(d) Actuators

Figure 1.5: Sensors and Actuators

1.4 Sensory Motor Integration

The two domains described above are vast and have been developed independently of each other. Though they are quite expansive and sophisticated a coordination between them is very important for an utilitarian robot that can be completely autonomous.

These two domains of robotics can be seen as analogous to the *sensory* and *motor* system in

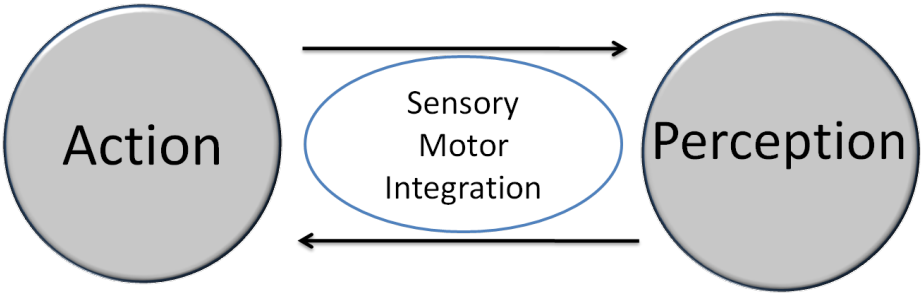

Figure 1.6: Sensory Motor Integration

humans and the *coordination* as *sensory motor integration*. This basically involves assisting the modules of the action space through the information from the perception space and vice versa. The lack of sophistication in this aspect is among the main hindrance in bringing the robots outside the laboratory confinement.

This thesis is a step towards enhancing the sophistication in the realm of *sensory motor integration* to augment the perception of objects in indoor environments. This is not a simple process which can be encapsulated to work as single module. It comprises of modules of both action and perception spaces which lead into and aid the functioning of each other. The different modules steer each other in a sequence leading towards accomplishing a required task. This is done by a process called “Bootstrapping” and it is detailed in the next section.

1.5 Bootstrapping

The term ”booting” in computing context refers to the process of incrementally updating the memory of the computer in order to run higher level processes over it. In robotic perception sense, it is the process of evolving the robots knowledge of its environment by *learning* the various aspects of it.

It is possible to bootstrap all the required information of its surroundings manually. But this is possible only in a particular, controlled environment and the robot cannot be useful in any new environment. Hence it is essential for the robot to be able to develop this knowledge on its own without external assistance.

The visual field is the most dominant among the different sources of perceptual stimuli. Similar to humans there are many aspects of comprehending the visual field. Among the various facets of visual perception, the focus of this thesis is on

Locating and identifying the objects, in order to update the robots knowledge of the environment.

Learning the objects appearance while it is being manipulated.

This can be relevant in context of a *personal robot* necessitated to accomplish a particular command like "Grab a Cola can". Though this seems a simple task for a human, it is quite a challenging task given the current level of technological sophistication. For the robot to follow this order, the environmental knowledge has to be accumulated. This involves

1. Identifying the table or support structures for the objects
2. Locating the different objects present
3. Recognizing the objects and identifying the required one
4. Augmenting the knowledge of the particular object by exploring it

Though these can be seen as individual process an underlying dependency of each process on the previous one can be seen. A sequence of independent modules that trigger the successive modules is essential for this. The modules essential to this process have been identified and sequenced as shown in the Fig.1.7. The individual modules can be best expressed with the following subjective terms

- Detect
- Recognize
- Track
- Grasp
- Learn

These can be compared to a human doing the same task. For example, if a person is required to "grab a cola from the table", he would first find the objects on the table and then identify the cola can from it and then grab it. Even in case when someone deliberately moves the object when he/she is trying to grasp it, the person moves towards the changed position of the object and grasps it.

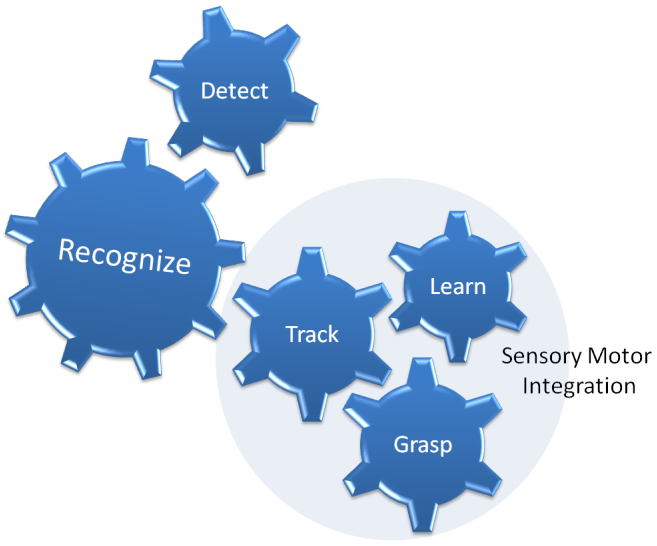

Figure 1.7: Learning Process

A similar approach can be implemented in robots as well where it *detects*, *recognizes* and then *grasps* the required object. Grasping an object which is deliberately moved during the grasping process can be tackled by continuously keeping track of the relative separation between the robot and the object and moving towards the object. During this process of grasping, the robot can see the object from different perspectives and this information can be used to enhance the robots understanding of the object. Through this a tighter sensory-motor integration is also achieved.

This learning process is illustrated symbolically in Fig.1.7 with a gear train mechanism describing the way in which one module leads to the other. The interaction between *tracking, grasping* and *learning* modules to achieve better *sensory-motor integration* can also be seen in the figure.

This thesis is organised as follows

- The following chapter of this thesis elaborates on the hardware configuration and the software platform on which the robot to test and evaluate the algorithms has been built.
- The subsequent chapters detail the modules of the process shown in the Fig.1.7
- A brief explanation of integration, future extensions to this work and conclusion complete this thesis

Chapter 2

System Configuration

"An ounce of action is worth a ton of theory."

(Ralph Waldo Emerson)

This excerpt from the renowned thinker is very relevant to research especially in the field of Robotics. Though a strong theoretical base lays the foundation for scientific progress, making it utilitarian in a practical manner drives forward the technological advancement. The deficits in theory can be identified by testing it in realtime conditions and this nourishes the theoretical knowledge by providing further motivation to enhance the sophistication. Though many established theoretical concepts exist to realize the tasks required by the modules explained in the previous section, it is important to implement them in practice to evaluate and improve them. Hence a robot has been developed to be a generic platform to evaluate and enhance theoretical aspects of both *action* and *perception* in a *personal robot* context.

Given the multitudinous means of uncertainties and disturbances, it is a challenge to construct an reliable, intelligent robot. A proper integration of hardware and software is required to build a effective robot. This requires a multi disciplinary team of researchers and a generic personal robot **"Robby"** [4], Fig.2.1 was built by the *Delft Robotics* team [1] of which I am a part of. The ideology of the team was to develop a low cost robot by making effective use of the hardware using smart control strategies.

Aimed at developing a generic personal robot, an analysis of the functionalities desired from the robot was done. The hardware was chosen based on these requirements and the software was developed to make the efficient use of the available hardware. A short description of the robots required functionalities is described below.

- The robot needs to visualize the world in spatial and chromatic dimensions. This is required for perception of humans, objects and gestures.
- A natural interaction of robot with the humans is required. It has to sense auditory information and also has to reciprocate through it.
- It has to move around and navigate in dynamic environments. It also needs to manipulate objects and reach for the objects at different locations.
- It has to sense and avoid the obstacles to ensure safety to both robot and the humans

As explained before a robust and reliable hardware is required to convert the environmental information into a form which can be digitally processed. The hardware chosen to satisfy these conditions is assembled to build the robot which has been used to test and improve the algorithms in this thesis. The subsequent sections detail the hardware and gives an overview of the software for the *Object Perception* modules.

[1] www.delftrobotics.nl

2.1 Hardware Description

The physical structure of the robot and the hardware and electronics is shown in Fig.2.1. A

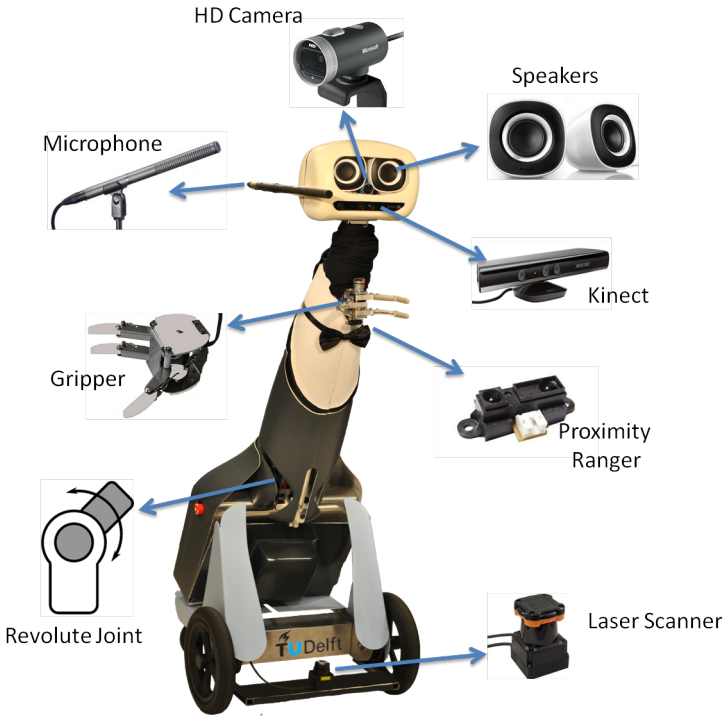

Figure 2.1: Robot Hardware Description

brief description of the hardware used is given below.

RGB-D Camera Classically only RGB camera's have been used for computer/robotic vision. Depth sensors like Laser Scanners and Stereo Camera's were used, but they were expensive, slow and not reliable. The Kinect developed by *Microsoft* apart from having a standard RGB camera, uses interpretation of structured light to estimate the distance. The *Depth Map* is synchronized with color map and it provides a RGB-D output in realtime speed of 30 Fps. This depth map is transformed into a Point-Cloud data using Projection matrices. By this, the environment is seen by the robot discrete points($[x, y, z]$ in m) containing RGB data.

HD Camera Though, the Kinect provides spatial and chromatic data of the environment, the resolution is not high for certain applications. Hence a *Microsoft LifeCam* has been used in conjunction to the kinect to get a High Resolution image for more reliable perception. The Kinect and the camera together are used sense the chromatic and the spatial aspects of the visual field of the environment.

Neck Joint The Camera system encapsulated within the robot head, has to be in different orientation for different circumstances like varying height of the person/table, etc. Hence a *Pan-Tilt* unit comprising of Dynamixels RX-64 is used.

Gripper The objects present in the everyday situations have varied properties like shape, texture, rigidity etc. It is quite complex if a standard fully actuated gripper is used as the finger configuration has to be computed for each and every circumstances. Hence an under-actuated gripper [15] developed in the Delft BioRobotics Lab which adapts itself to different object properties has been used to simplify computational complexity.

Proximity Sensor Since an adaptive gripper is used, the only feedback to control the gripper is the presence/absence of the object. Hence a SHARP(0A41SK F15) proximity sensor is used to detect the presence of object within the manipulable range of the gripper. The configuration of Gripper and the Proximity sensor is shown in the Fig.2.2. This setup proved to be very useful for reliable object manipulation explained in the later chapters.

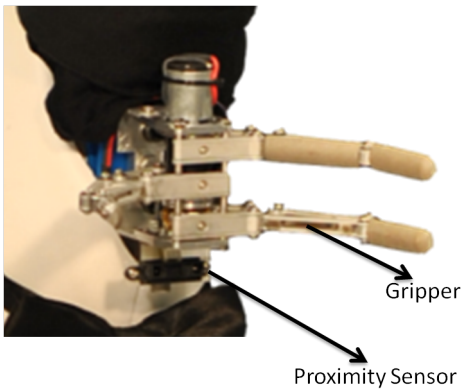

Figure 2.2: Gripper-Sensor Configuration

Hip Joint A Revolute joint in the hip is the only way to vary the height of the gripper/head to adapt to different conditions.

Base A mobile base consisting of two wheels coupled to 12V/3.6A DC motors is the platform over which the entire system has been developed. The Motors in the base and the hip joint are controlled by 3Mxel, an advanced Motor Controller developed in the Laboratory [4].

Microphone and Speakers A highly directional (Shotgun Audiotechnica AT-8035) Microphone is used to robustly receive the speech in noisy real-time conditions and a Philips (SPA2201) speaker are used for the natural Human robot interaction.

Laser Scanner A planar laser scanner (HOKUYO URG - 04LX) with a range of 4m with a precision of 1mm is used along with the mobile base to navigate in dynamic environments avoiding the obstacles.

The hardware being identified and the robot being constructed physically, intelligence is provided by the software which is described in the next section.

2.2 Software Overview

A robust software has to adapt to varied and dynamic working conditions, especially in the context of personal service robots. A generic software framework on which different modules can be developed for the ease of interaction between them is the backbone of robots intelligence. A large scale open source project *Robot Operating System* abbreviated *ROS* is used as a platform and some important aspects of this include

- Versatile C++ programming API
- Integrated drivers for interaction with a wide variety of robot hardware
- Shared memory for easy interaction between multiple processes
- A generic visualization tool to facilitate user interpretation of manipulated data
- Debugging tools to understand the interaction between multiple processes and identify faults.

Having a reliable software platform, the independent modules can be built on top of it. The overview and a brief description of the software modules specific to this thesis is given below. The necessary modules for the process explained in the previous chapter, Fig.1.7 are analysed objectively and fragmented into sub-modules which can be independently tackled. These are mapped on to a hierarchical structure as shown in the Fig.2.3 This hierarchy tree shows the overall problem is technically approached step by step. A brief description of each of the modules is given below:

3D Object Localization The 3 dimensional location of the objects relative to the robot frame has to be determined before it can be analysed. This module locates the prospective objects in the scene.

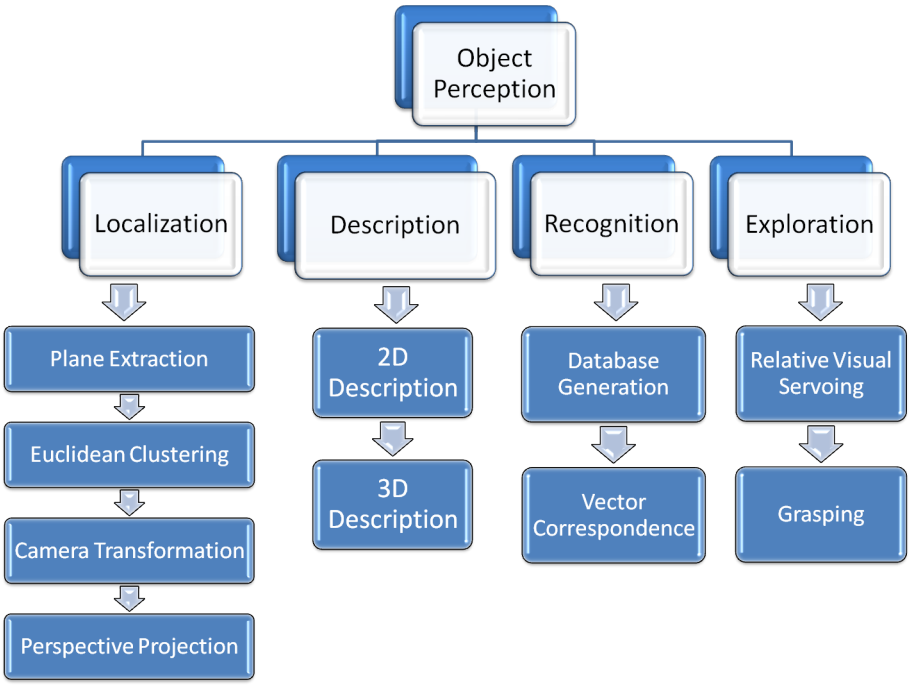

Figure 2.3: Sub Module Hierarchy

Description Once the object has been located, its chromatic and geometrical properties are mathematically characterized by a Feature Vector.

Recognition The query object in the scene is recognized as known or unknown by comparison with the database of previously known objects.

Exploration The object has to be learned by obtaining multiple views from different perspectives.

Tracking The object also has to be tracked continuously while it is being explored. This also assists in manipulation of the objects.

Manipulation The objects can be selected and manipulated depending on the contextual requirements in different scenarios. It can be manipulated to obtain a different view point for learning more features or it just has to be grasped to be delivered in a different location (for a general purpose robot).

The 3D Object localization has been implemented using the *PointClouds Library*, the Tracking uses *OpenCV* and Description and Recognition has been implemented in *MATLAB* environment. The communication between *Matlab* and *ROS* has been achieved using the IPC(Inter-Process Communication) bridge [2]. The ensuing chapters elaborate on the sub modular details of the major modules.

Chapter 3

3D Object Localization

Detecting the object in the scene is the first step triggering the entire bootstrapping process as depicted in the Fig.1.7. Technically *detecting* can be denoted as *3D Object localization*. The objective of this module is to locate the objects present in the visual field of the robot. This can be performed in two domains.

1. 2D - Chromatic information
2. 3D - Spatial information

The *Feature Integration Theory* for human visual search [26] defines the term **"Saliency Map"**. It concludes that visual input is first decomposed into smaller feature maps. Spatial locations in which the local features stand out from their surroundings combine together to generate the Saliency map.

Chromatic Object Location

Cameras capturing the colour information of the scene into a 2 dimensional image have been prevalent for quite long and hence many algorithms to locate the objects based on this information have been developed. A research into human visual search mechanisms ([26]) lead to the *Feature Integration Theory* which introduces the term **"Saliency Map"** which contain locations in the visual field that stand out contrastingly from their surroundings. Taking inspiration from this, saliency based localisation methods have been developed. Zhang et al. [13] showed from the analysis of natural image statistics that statistical singularities in the spectrum of the image are responsible for the anomalous regions in it. They developed methods that analyse the spectrum of gray-scale images. This idea was further developed by Rudinac et al. [20] to incorporate the available color information to enhance the detection.

Though these methods are sophisticated, the robustness to be suited for dynamic environments is lacking. It can be attributed to the non-availability of the spatial aspect of the visual field.

Spatial Advantage

When the additional information about the spatial distribution of objects in the scene is available, it becomes more easier to robustly distinguish objects from the background. The difference between a *chromatic* and a *spatio-chromatic* image of the same scene can be seen from the Fig.3.1. For the human eye the objects can be easily identified from both the images. This is because human visual interpretation system is sophisticated to comprehend spatial structure from various aspects like *texture*, *shading/shadows*, *scale*. But for the robot, a colour image is just a sequence of numbers representing the colour. The spatial image provides additional information of the points$[x, y, z]$ in space representing the corresponding colour. The spatial location of the points is available only because the *distance*(depth) of each point from the camera can be identified. This adds another dimension to the sensory data and this can be used quite well to locate the objects in the scene.

While *stereo-camera* systems were the most common means of obtaining spatial information in the past, they were quite expensive and the depth information was not very reliable.

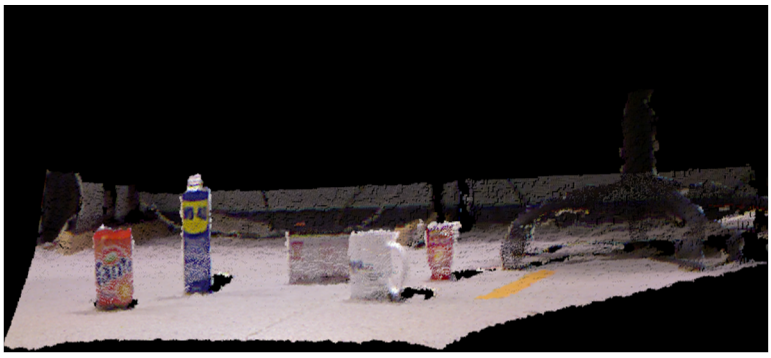

Figure 3.1: Spatio-Chromatic Image

Hence the localization methods based on spatial data were not established in the past. But recently with the advent of *Kinect* from Microsoft, a reliable depth information in indoor environments is afford-ably available and many algorithms have evolved and become established in the recent years to exploit the additional dimensions. The process of locating the objects in 3 Dimensional space is explained below.

3.1 Modules of 3d Segmentation

As explained earlier, the Kinect provides a 3 dimensional map of the scene with RGB data at a rate of 30 frames per second. The methodology of processing this information to identify the objects in the scene and determine their 3D($[x, y, z]$) relative position from the robot is termed *3D Object Localization*. This consists of different sub processes like

- The planar regions are generally the support surfaces for the objects. These regions have to be first identified.
- After eliminating the points belonging to the major planar regions, the rest of the sparse points have to be clustered to convert the point cloud to set of points belonging to a particular object
- The location of these objects in the 2D color image have to be obtained for further processing of the object's appearance for recognition.
- As the resolution of the image provided by the Kinect is not sufficient for object recognition, a High Definition camera is used in conjunction and hence the object location has to be transformed into the image from the camera.
- The image of the scene is now converted into a set of images corresponding to the objects.

The method of achieving the above mentioned task is further explained in the next sections.

3.2 Plane Extraction

In a normal indoor living environment, most of the support and structural elements like Walls, Tables, Floor are *Planar*. This can be used as an advantage as the robot cannot affect these larger structural elements. Eliminating these regions from the sensory data will expedite the clustering process as the number of points to be processed is reduced. Hence *plane extraction* is the first step in object localization.

Analytically a Plane in 3 space can be represented by Eq.3.1

$$Ax + By + Cz + D = 0 \tag{3.1}$$

where, $[x, y, z]$ in this equation represent the spatial coordinates of the points in the scene. The process of identifying the points that satisfy this relation would be straightforward if the parameters $[A, B, C, D]$ of the plane are known. Though linear regression methods could be used for this purpose, there are other associated issues like

- Noise
- Large number of Outliers which is a consequence of noise

Since these are associated with the hardware and cannot be avoided, a software solution is required for this problem. This problem can be solved by using a *Probabilistic Approach* instead of a *Deterministic* one. A method called Random Sample Consensus, abbreviated as RANSAC [11] is used.

RANSAC

RANSAC is an iterative method which determines the parts of data corresponding to the required parametric form probabilistically. The accuracy of the estimated solution increases with the number of iterations which directly influences the computational time. Hence there is a trade-off between the accuracy required and the time feasible. An explanation of this estimation process for a simple case of *line extraction* is given below. This process can be easily extended to incorporate a single dimensional higher *plane extraction*. The points that belong to the model which has to be determined are termed *inliers* and the other points present in the dataset that don't constitute the model are termed *outliers*.

A line in 2Dimensions can be represented in the parametric form

$$Ax + By + C = 0$$

Given a set of points in 2D, the objective is to estimate the points which constitute a line as illustrated in the Fig.3.2. The inliers and outliers are also illustrated in this figure. A

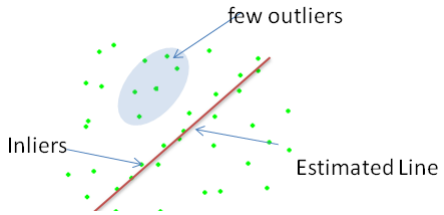

Figure 3.2: RANSAC for Line Model

simplistic algorithm proceeds iteratively and at each iteration

- A minimum number of points required to fit the parametric model is randomly selected. In case of line the minimum number of points is 2 and for a plane it is 3.
- Considering these points as *inliers*(hypothetical), the parametric plane model which fits these points is reconstructed.
- All the other points in the data are validated with this obtained model and when the points fit well to the reconstructed model, they are also considered hypothetical inliers. The points fit well, if the perpendicular distance of the points to the line/plane is less than the parameter **distance threshold**.

- The model is considered to be reasonable if a sufficient amount of inliers are determined. This is specified by the parameter **Minimum number of inliers**.
- If the model is deemed reasonable, it is refined by re-estimating the model over the set of located inliers.

At each iteration the generated model can either be rejected due to lack of sufficient number of inliers or refined through estimating over a large set of identified inliers. There might be cases where the process can iterate indefinitely due to the iterations not having enough inliers. A parameter **Maximum Iterations** is specified beyond which the algorithm terminates without estimating any required regions in the data.

It can be seen that there are many parameters involved in this process. They are summarized below

Maximum Iterations The maximum number of iterations after which the algorithm terminates. This is set as 100 to avoid long computational overhead. This has proven to be sufficient for estimating the planes from the Kinect point cloud information.

Distance Threshold to the model The maximum distance of a point from the estimated plane for it to be considered an *inlier*. This is a measure of noise tolerance in the data. A distance of 2cm has been found to be adequate.

Minimum Number of Inliers A minimum number of points that satisfy an estimated model, for the model to be deemed valid. This parameter being set at 400 to avoid estimation of certain noisy data as planar structures.

The algorithm locates planes starting from the largest plane. Each time a plane is located, the points belonging to this plane are eliminated from the data source and the algorithm is run over this reduced data again. The planes are eliminated from the data till

$$Points_{remaining} \geq 0.3 \times Points_{initial}$$

The value of 0.4 was found to ensure that while the larger planar structures are eliminated, small planar elements that could be prospective objects still remain in the data.

The Fig.3.3 shows the theoretically expected results from a sample dataset. The Fig.3.4

Figure 3.3: Determined Planes

summarizes the entire plane extraction process.

Figure 3.4: Summary: PlaneExtraction

3.2.1 Euclidean Clustering

Once the larger planar components are eliminated from the data source, the objects have to be located in from the remaining point-cloud. This is done by the process of *Euclidean Clustering*

This process can be seen as a *nearest neighbour search* in 3 dimensions or analogous to recursive floodfill starting from different points until all the points are processed. The process is quite simple in which from each point in the dataset, points located within an euclidean distance of *clusterthreshold* from the seed point are agglomerated as a cluster. The euclidean distance between two points in 3 dimensions is given by

$$D_{12} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

The *threshold* implies two objects placed within this threshold distance of each other will be perceived by the robot as a single object. But when this threshold is reduced, parts of same object can be seen as different clusters. For example the handle of a mug will be considered different objects. Hence a tradeoff between these two conditions after evaluating on different everyday used objects is used as $5cm$.

This process continues until all the points in the dataset are assigned to a particular cluster. Due to the noisy nature of the sensor data, there might be clusters which have very few points. Also some larger structures in the scene like chairs etc are clustered together as an object. It is not possible for the robot to act over such larger objects. Hence limits on object size are imposed. The object size is quantified by the number of points in a cluster. The limits used are

- Minimum Cluster size: 100
- Maximum Cluster size: 500

The objects clustered from the image in the Fig.3.1 are shown in the Fig.3.5. The points constituting a single object are coloured similar in the illustration. This *clustering* process

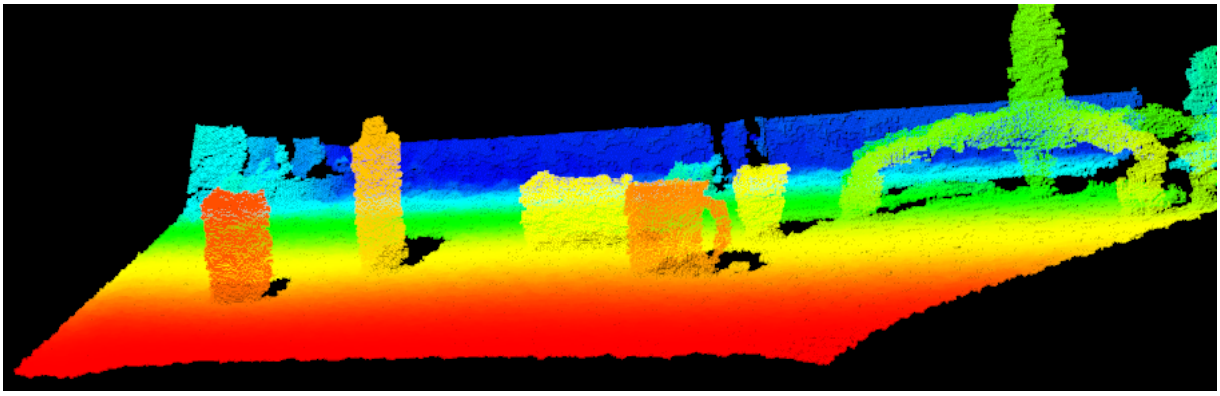

Figure 3.5: Clustered Objects

provides sequence of cluster of points that represent the objects identified in the scene and is summarized by the by Fig.3.6. The points obtained are in real world co-ordinates

Figure 3.6: Summary: Clustering

represented by $[x, y, z]$ in m. The RGB image corresponding to the objects has to be obtained for chromatic description to be used in recognition in later stages. This is done by the process of 2D projection and image transformation as explained below.

3.2.2 Image Transformation

The clustered pointclouds have to be projected into 2D pixel plane in Kinect frame to obtain the object boundary in pixels. But this image has a resolution of only 640×480 pixels. Though this resolution is adequate for spatial(depth) processing, it does not have sufficient data for processing of RGB information for object recognition. Hence an image from the HD Camera has to be used for object appearance data.

This is done by first obtaining the object boundary in the Kinect image and then mapping it to boundary coordinates in the camera image.

Problem

Obtain the object location in Kinect image and determine a linear transformation from any point in Kinect Image to the corresponding point in the camera image

Assumption

The distance (z) to any point in the scene from optical centre of Kinect and camera is the same. This assumption is credible as the image planes of both the sensors lie in the same $x - y$ plane.

Solution

- Perspective Projection of cluster points in Kinect frame.
- Rigid Transformation of points corresponding to the object boundary to camera frame
- Perspective Projection of transformed points into the camera image plane

Camera Transformation

The Spatial configuration of the Kinect and Camera being rigid, a linear transformation can be determined between them. This is done by the process of *Stereo Calibration*. This has been done with the help of calibration toolbox for matlab as in [5], which is based on the standard calibration algorithm proposed by zhang [30].

Once the system is calibrated, the *intrinsic parameters* of the individual camera's and also the *extrinsic parameters* which quantify the geometrical relationship between them. While the intrinsic parameters are used for *perspective projection*, the extrinsic parameters define the *transformation* of point-cloud between the camera frames.

intrinsic parameters

The intrinsic parameters can be represented as a matrix as in Eq.3.2.

$$A = \begin{bmatrix} F_x & \gamma & C_x \\ 0 & F_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Where f_x and f_y are the focal lengths in x and y directions, c_x and c_y are the principal points(image centres) in x and y directions. γ is the distortion coefficient.

After the calibration process, these parameters have been identified as in Eq.3.3

$$A_{camera} = \begin{bmatrix} 990.2451 & 0 & 670.8475 \\ 0 & 989.2745 & 418.1278 \\ 0 & 0 & 1 \end{bmatrix}, A_{kinect} = \begin{bmatrix} 521.1204 & 0 & 313.4031 \\ 0 & 520.5478 & 256.1400 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Extrinsic Parameters The extrinsic parameters specify the geometric transformation between the camera centres in real-world co-ordinates. This can be represented with a **Rotation Matrix** R and a **Translation Vector** T as in Eq.3.4

$$R = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{bmatrix}, T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3.4}$$

After calibration, these parameters have been identified as in Eq.3.5

$$R_{Kinect2Cam} = \begin{bmatrix} 0.9996 & -0.0123 & -0.0262 \\ 0.0125 & 0.9999 & 0.0072 \\ 0.0261 & -0.0075 & 0.9996 \end{bmatrix}, T_{Kinect2Cam} = \begin{bmatrix} 14.7016 \\ 40.8244 \\ 0.6894 \end{bmatrix} \tag{3.5}$$

Once the extrinsic parameters are identified, the Point Cloud can be transformed from the Kinect frame to the Camera frame. If X_c and X_k are the real-world coordinates of a particular point in the Camera and the Kinect frame, they are related by the transformation as in Eq.3.6

$$X_c = R_{Kinect2Cam} * X_k + T_{Kinect2Cam} \tag{3.6}$$

3.2.3 Boundary Determination in the Kinect Image

The object boundary in the Kinect image is obtained as follows.

- Each point in the cluster is projected into a x and y co-ordinates in the kinect frame using the $[x, y, z]$ data and also the Intrinsic Parameters using (3.7).

$$Pixel_x = \frac{X_{point}}{Z_{point}} F_x + C_x \tag{3.7a}$$

$$Pixel_y = \frac{Y_{point}}{Z_{point}} F_y + C_y \tag{3.7b}$$

- The boundary is obtained by selecting the Minimum and Maximum Pixel positions in both x and y directions.

Perspective Projection

When the spatial location of a point is available in the camera frame, the camera model which is known from the intrinsic parameter calibration can be used to get the **Pixel** position in the camera image through the Eq.3.7, where the $[X_{point}, Y_{point}, Z_{point}]$ are in the camera frame.

It is possible that the entire point cloud can be transformed from Kinect frame to the Camera frame and the *Perspective Projection* has to be performed only once. But applying the transformation for all 3D points takes computationally longer time, and hence transforming and projecting only the 4 points corresponding to the boundary is faster.
Hence, if X_{kinect} and Y_{kinect} are the coordinates in the Kinect image and Z is the depth at this point, the X_{camera} and Y_{camera} in the Camera image are obtained as

$$X_{kw} = \frac{X_{kinect} - C_{kx}}{F_{kx}} Z \tag{3.8}$$

$$Y_{kw} = \frac{Y_{kinect} - C_{ky}}{F_{ky}} Z \tag{3.9}$$

$$\begin{bmatrix} X_{cw} \\ Y_{cw} \\ 1 \end{bmatrix} = [R \mid T] \begin{bmatrix} X_{kw} \\ Y_{kw} \\ Z \\ 1 \end{bmatrix} \tag{3.10}$$

$$X_{camera} = \frac{X_{cw}}{Z} F_{cx} + C_{cx} \tag{3.11}$$

$$Y_{camera} = \frac{Y_{cw}}{Z} F_{cy} + C_{cy} \tag{3.12}$$

$$\tag{3.13}$$

All the parameters used are obtained in the calibration process. The RGB image of a scene, the visualization of the point-cloud data, the segmented objects in the kinect image and the transformed objects in the camera image are shown in the Fig.3.7.

It can be seen that, the explained sequence of methods provide good segmentation of the objects from the scene. But due to the limitations of the Kinect hardware, certain objects which are

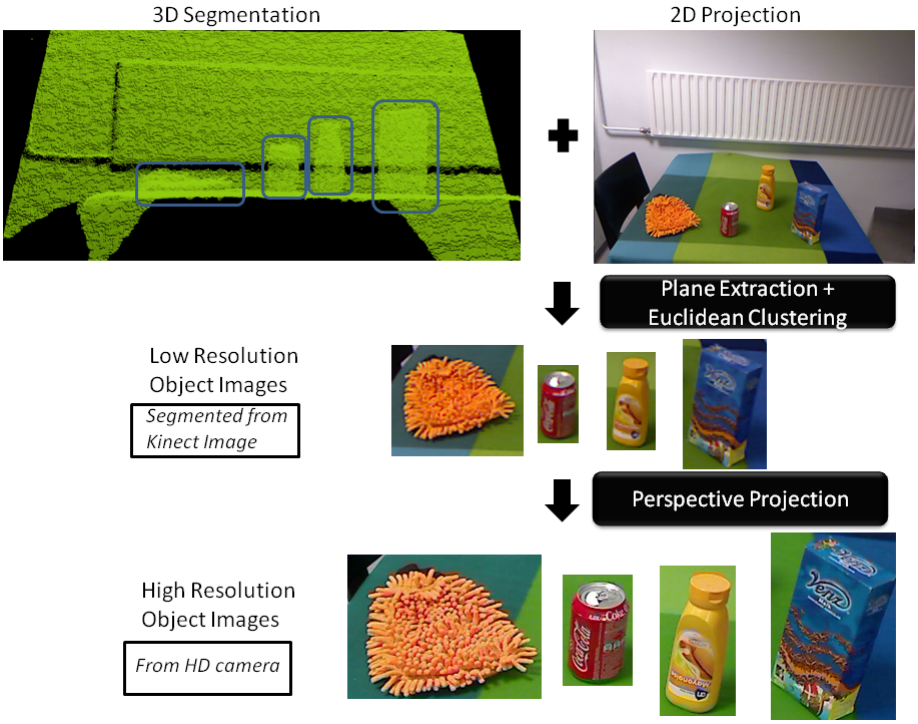

Figure 3.7: Summary: Localization

- Black
- Transparent
- Non-Reflective

cannot be captured in the depth dimension. Hence a combination of saliency based methods [20] and this 3D segmentation process in the future should yield robust segmentation in dynamic and also outdoor environments.

Now the chromatic image of the object candidates being available, they have to be mathematically described for recognition. This extraction of features and their interpretation for recognition is elucidated in the next section.

Chapter 4

Description and Recognition

4.1 Description

Once the candidate objects are located in the scene, they have to be described mathematically for the robot to comprehend it. These descriptors are used to detect novel objects and to recognise the previously learned ones. It can be observed in humans that different people remember the same object with different characteristics. Though it cannot be analytically determined how objects are represented within the brain, it can be seen that some people can very quickly recognize objects or faces while some take longer time. Similarly different methods of description have been developed in the field of Object Recognition in Computer vision. They can be broadly grouped into

- Local Descriptors
- Global Descriptors

Local Descriptors:
As explained before, these involve locating salient points in the images which are very stable and then describing these images based on their local feature properties like gradients. A widely used local descriptor is the Scale Invariant Feature Transform (**SIFT**) was proposed by Lowe in [8]. This transform identifies scale and rotation invariant key-points in an image and describes every such point by a vector of length 128. The SIFT feature points have also proven to be quite robust to viewpoint change (until 40%).

Though this operates only with grayscale images, there have been improvements to include colour information with local key points like in [9]. Object recognition using these methods involves comparing every keypoint and clustering the most matching points together. This has to be done on all the source images in the database and it leads to a very large increase in **computation time** with increasing database size thereby making them not suitable for realtime implementation. Hence a Global descriptor based approach as explained below has been used

4.1.1 2D Global Feature Vector

It has been shown in [19] that a unified global descriptor that encapsulates the color, texture and gradient distribution information into a single Feature vector of dimension 252. In this descriptor,

- The colour is quantified through a histogram in the HSV space
- The texture is represented by the properties of the Gray Level Co-occurance Matrix (**GLCM**)
- The gradient distribution is quantified by a histogram of the edge orientations obtained through a Canny edge detector.

This type of feature vector is shown to provide good robustness to common issues like

- Illumination Variance
- Occlusion
- Noise
- Rotation

By this process, the raw RGB image of the object is transformed into a 252 dimensional feature vector. This can be seen as analogous to projecting the image into a lower dimensional space or a sort of PCA(Principal Component Analysis). This process can be visualized as shown in the Fig.4.1. It can be seen from the Fig.4.1 that while the features of the Tea Box

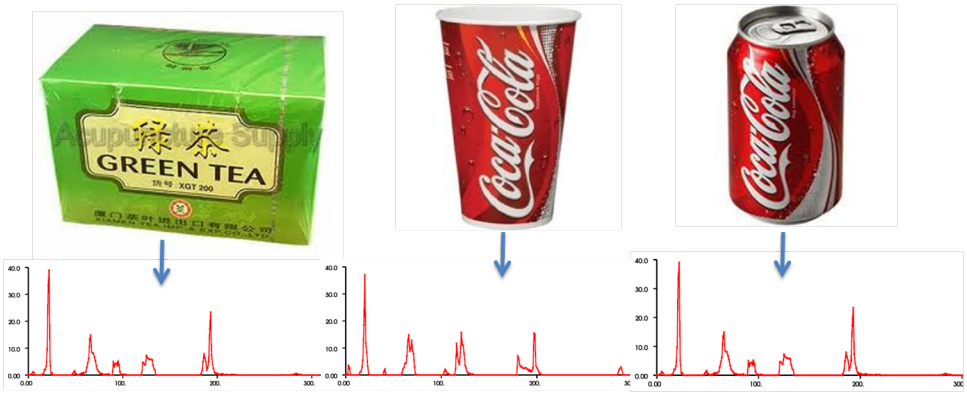

Figure 4.1: RGB Feature Descriptor

is quite different from that of a Cola can, the features of the Cola can and a Cola cup are almost the same. These kind of situations will make the robot believe that the Cola cup and Cola can are the same objects. This is because only Chromatic information is used to describe the object. Hence the use of 3D global feature vectors was proposed as a possible solution to this problem.

4.1.2 3D Global Feature Vector

Though many keypoint descriptors have been proposed in literature, a global descriptor is more desirable as explained earlier. It is also easier to combine this with the already existing 2D description techniques.

A *VFH*, Viewpoint Feature Histogram which represents the geometric interrelation between the points that constitute an object has been developed in [22]. This VFH is a vector of 308 dimensions which consists of two components

- A viewpoint direction component
- A surface shape component

The surface shape component consists of a histogram of relative pan, yaw and tilt angles between the point pairs.

The pan(α), tilt(ϕ), yaw(θ) between pair of 3D points p_i, p_j and whose surface normals are n_i, n_j is

$$\begin{aligned} \alpha &= v.n_j \\ \phi &= \frac{p_j - p_i}{d} \\ \theta &= \arctan(w.n_j, u.n_j) \end{aligned}$$

where u, v, w are the Darboux frame coordinate system at p_i. The angles are calculated between every pair of points in the point cloud of the object and a histogram is obtained for each of these angles by using 45 bins for each angle. Also the euclidean distance between the centroid and each point is determined and a histogram of 45 bins is calculated. These

two things together constitute the surface shape component which is of length 180 (45×4). This constitutes the surface shape component.

The viewpoint component is the histogram of the angle made by the surface normals at each point to the *central viewpoint direction* quantized into 128 bins. This central viewpoint direction is the direction of the surface normal at the centroid of the pointcloud comprising the object.

Concatenating these two components together yields a vector of dimension 308 (280+128), which is termed as *Viewpoint Feature Histogram*.

This 3D feature vector of different objects will resemble the visualization in Fig.4.2. It

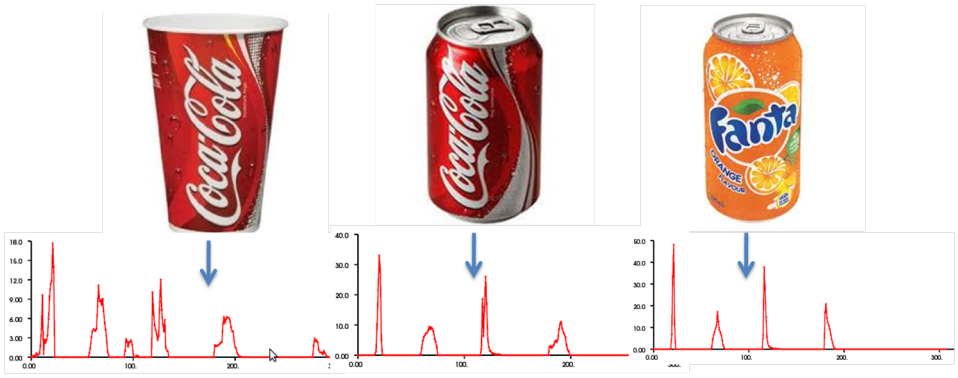

Figure 4.2: 3D(VFH) Feature Descriptor

can be seen from the Fig.4.2 that the 3D Feature for the Cola can and cup are quite different. But this feature vector cannot distinguish similarly shaped objects. It can be seen from the Fig.4.2 that, the Fanta can and Cola can have similar feature vectors and hence the robot will believe they are the same kind of objects.

Hence the solution to the problems with 2D and 3D feature vectors is to combine them together as a unified 2D+3D global descriptor, instead of using them disjointly.

4.2 Recognition

Object recognition is a two step process. A database of the objects to be learned by the robot is created first. Then, every object that needs to be recognized can be vectorially compared with the existing database.

The database of sample images consisting of multiple views of each object is created manually and is used later for recognition. The recognition algorithm used by Rudinac et al. in [19] is extended to incorporate the additional 3D information available on the objects. Entire merging procedure is extensively described bellow:

Similar to Rudinac et al. [21] every segmented viewpoint of the object is described using both 2D and 3D visual features. Since the objects appearances can vary significantly both in color, texture and 3D shape, all corresponding feature vectors are extracted and dominant features of the object automatically calculated.

For this previous research on fast and robust feature descriptors [19] as explained in section 4.1.1 was utilized. A color histogram including hue, saturation and value used as a color descriptor, a Gray Level Co-occurrence Matrix (GLCM) as a texture descriptor, and an edge histogram for 2D shape information, all combined into a single $D1 = 256$ dimensional vector $\mathbf{d}$. This feature vector thus describes both textured and untextured objects.

As it is shown in the Figure 4.1, several objects can have very similar texture but different shape.Hence to describe the 3D shape of the object, VHF histogram as explained in section

4.1.2 is added to the description as a feature vector $\mathbf{v}$ of the dimension $D2 = 308$. These two feature vectors are now concatenated into a single feature vector $\mathbf{f}$ of dimension $D = 564$.

Since some features might be more descriptive than others, a normalization step is performed in order to emphasize the dominant features. Such normalization has advantages over methods that treat every element in the feature vector as equally important as it was shown in [19].

Now all viewpoints of all objects from the learning set are merged in one feature matrix, $\mathbf{F}$, where $\mathbf{F}(i,j)$ is the value in row i and column j, which gives the j-th feature of feature vector $\mathbf{f}_i$. As the different features are obtained in a different way, the columns in the feature matrix have significantly different values. So a first normalization is performed on $\mathbf{F}$ by dividing all values by the maximum value in their respective column:

$$\bar{\mathbf{F}}(i,j) = \mathbf{F}(i,j)/\mathbf{m}(j) \tag{4.1}$$

$$\mathbf{m}(j) = \max_{i=1}^{N} \mathbf{F}(i,j) \tag{4.2}$$

where $\mathbf{m}$ is the vector with the maximum values per feature, N is the number of observed feature vectors in the matrix, and $\max_{i=1}^{N}$ gives the maximum over all rows.

The second step in the normalization procedure emphasizes dominant features. The dominance of each feature is captured in the weight vector $\mathbf{w}$, which is calculated using the variance over all views of the object:

$$\mathbf{w}(j) = \frac{1}{m_j} \log_2 \left(\frac{1}{m_j} \operatorname*{std}_{i=1}^{N} \bar{\mathbf{F}}(i,j) + 2 \right) \tag{4.3}$$

$$m_j = \frac{1}{N} \sum_{i=1}^{N} \bar{\mathbf{F}}(i,j) \tag{4.4}$$

where $\operatorname{std}_{i=1}^{N}$ calculates the standard deviation over all rows. The feature dominance is then used to reweigh $\mathbf{F}$, so that more dominant features get emphasized:

$$\hat{\mathbf{F}}(i,j) = \bar{\mathbf{F}}(i,j) \cdot \mathbf{w}(j) \tag{4.5}$$

This dominance weighting makes the object descriptors more robust to changes in viewpoint and illumination conditions, allowing more robust object recognition.

Now in the case of a real scene, all objects in the scene are first localized and in order to recognize them described using proposed joint vector. An object is described using feature vector $\mathbf{g}$ and normalized using stored weight vectors normalization vectors $\mathbf{w}$ and $\mathbf{m}$:

$$\hat{\mathbf{g}}(j) = \mathbf{g}(j) \cdot \frac{\mathbf{w}(j)}{\mathbf{m}(j)} \tag{4.6}$$

The normalized feature vector $\hat{\mathbf{g}}$ is then matched to the stored dominance-weighted feature matrix, $\hat{\mathbf{F}}$, and the distance to each of the feature vectors in the matrix is determined using the L1 distance:

$$\mathbf{d}(i) = \sum_{j=1}^{D} \|\hat{\mathbf{g}}(j) - \hat{\mathbf{F}}(i,j)\| \tag{4.7}$$

The object with a smallest distance is then considered as a recognized object. For the detection of the unknown objects, please refer to Rudinac et al. [21].

Table 4.1: The results of object recognition in conditions of uniform illumination

Descriptor	Precision(%)
2D Features without normalization	81.30
2D Features with normalization	86.60
2D + 3D Features without normalization	87.71
2D + 3D Features with normalization	91.55

This extended algorithm has been evaluated on a subset of the standard RGBD dataset available [17]. This dataset consists of 300 commonly used household objects which are organized into 51 categories. This dataset is very relevant as the focus of this thesis also is on objects in indoor (household) environments. To evaluate the recognition algorithms, the database was divided in the training set containing random 10% of all viewpoints of the objects and the test set containing rest. The training dataset is small to mimic the real-world situation where only limited number of object instances in available for learning. Both 2D descriptor and joint 2D+3D descriptor were tested with and without normalization step and results are displayed in table: 4.1. From results we can draw following conclusions:

1. Normalisation step improves the recognition performance in both cases
2. Adding 3D descriptors significantly improves the performance
3. Recognition performance of over 90 % is sufficient for the method to be applicable for service robots

One addition to this module is classification of the objects in the scene as known or unknown objects and application of recognition for online learning of objects, which is explained in more details in [21]. Depending on the context, the robot either has to grasp a known object or explore to learn more about the unknown object. In this thesis the method used in [21] is extended and a generic grasping method which also learns the object model while it is being manipulated is proposed, as explained in the next section.

Chapter 5

Active Object Manipulation

The need for action-perception coordination as explained in a earlier chapter deals with acting on the environment based on the perceived information. The previous chapters on *object localization* and *recognition* focus on perceiving the environmental data. This chapter on *object manipulation* deals with grasping an object(acting on the environment) based on the perceptual data.

Grasping is a complex task which depends not only the modules in previous chapters(perception space), but is influenced by many other factors like

- Properties of the object
- Location of the object
- Characteristic of the robot

Also as in humans, manipulating an object by a robot always consists of two distinct phases

1. Approach phase
2. Grasp phase

The *approach phase* consists of moving the end-effector to a position from which the object to be grasped is within its vicinity. The *grasp phase* deals with obtaining the appropriate end configuration of the gripper phalanges so that the object is within the gripper.

While the *grasp phase* is influenced by the properties of the object, the *approach phase* depends on the location of the object and also the characteristics of the robot.

Properties of the object

The everyday objects differ widely in their properties like shape, size, weight, fragility, etc. Objects like a ball are spherical, cans are cylindrical, boxes are cubical while certain objects like wine glass or an apple are obliquely shaped. Each of these objects can have varied sizes from a small ball to a large box and also the weight and fragility depends on the material of the object. While a paper or a plastic cup is lighter and deformable, a metallic or porcelain mug is heavier and rigid. These wide variations in object properties significantly influence the grasping strategy.

Different gripping configurations are required for differently shaped objects. There have been many methods proposed to determine the final gripping configuration based on the visual information processing. A few of these methods can be summarized as:

- All the possible objects in the workspace of the robot are learnt offline. A stable grasp configuration for every object and an individual pose is obtained by verifying the final grasp configuration in the simulator. This method is specific for an anthropomorphic arm [3] and works only for fixed pre-learned objects.
- Richtsfeld et al. [18] use an additional laser scanner to identify planar surfaces on the top of the object and generate grasp configurations such that the centre of mass of the surface lies within the final contact points of the gripper with the object

- Another approach was to combine the information from the wrist camera and a line laser and extracting object silhouette from multiple viewpoints [12]. A parallel jaw gripper being used can best grasp approximately flat parallel surfaces. Hence these type of surfaces are identified in the object and the final gripper position is determined by using robust force closure criteria in the simulation.

Apart from estimating the final gripping configuration, determining the force applied on the object is still a difficult task as this cannot be easily estimated from the available sensory information. This is critical as applying a larger force on an object like plastic cup can deform the object whereas applying a smaller force on a porcelain mug can cause insufficient grip and also slipping of the object after gripping causing damage to it.

Though many vision based solutions have been proposed for grasping, it is very difficult to reliably grasp previously unknown objects. This problem has been approached from a mechanical perspective in the Delft Biorobotics Laboratory where an under-actuated, adaptive gripper has been developed [15]. This Delft Hand[1] has three phalanges controlled by a single motor and the configuration automatically adjusts to the objects with different properties. Hence this is chosen as an end effector to eliminate the computational complexity involving determination of finger configuration of the final grasp. With the problem of grasping objects with different physical properties being addressed by a mechanical solution, reaching the different locations where the objects can be present has to be analysed.

Location of the object and robot characteristics

The influence of object properties on the *grasp phase* was discussed and the dependency of the *approach phase* on the location of the object and the characteristics of the robot is discussed below. The objects to be grasped can be located at different positions like

- Table of various heights
- Floor
- Refrigerator or Cupboard shelves, etc

A different trajectory planning is required for different locations of the object. The currently existing techniques proceed in an approach as illustrated in the Fig.5.1. Advanced methods

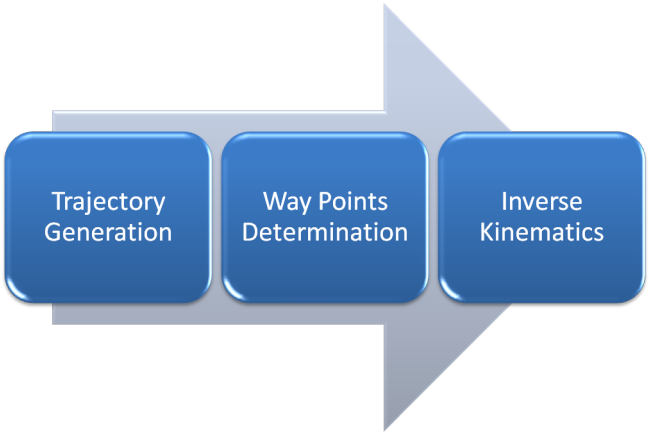

Figure 5.1: Existing Grasping Methodology

like Rapidly-exploring Random Trees (RRT) [16] and Kinodynamic motion planning [25] are available to generate the trajectory of the end effector. The motion of the individual joints in the arm is obtained from the inverse kinematics which are implemented in open source libraries like KDL[2] and OMPL[3].

[1] www.lacquey.nl
[2] http://www.orocos.org/wiki/orocos/kdl-wiki
[3] http://ompl.kavrakilab.org/

Even though this approach has been studied, refined, developed and is in a quite sophisticated state now, there still is an issue with the fundamental aspect which is *"Open Loop Operation"*. The open-loop aspect comes from the fact that once an object is located, the grasping sequence represented in Fig.5.1 is initiated. This executes the arm joint trajectory and this can fail in the situations where

- The object is slightly moved by disturbance to support elements like a table
- The object is intentionally moved by external factors
- There is an offset due to joint sensor encoders.

When the object is moved, the end effector reaches only the initially planned position and the gripper tries to grab the non-existing object. This leads to failure and reduces the reliability of the grasping. The problem is inherent to this type of approach paradigm and cannot be solved by algorithmic improvements.

Alternate techniques based on *closed loop* paradigm have been proposed in [14] and [6]. Jafari et al. [14] proposed the method of grasping based on relative visual servoing in the year 2004. Though they suggested that visual feedback can be beneficial to the grasping process it was concluded by them to be not feasible as it required continuous visual feedback. This was due to the lack of algorithmic sophistication and the fast and affordable computational power at that time.

The method proposed by Calli et al. [6] is quite advanced and manoeuvres the gripper to a position most suitable for grasping. This is achieved by reaching the viewpoint which provides the maximum curvature region of the object to be grasped. This method uses elliptic fourier descriptors (*EFD*) calculated from the silhouette of the object to quantify the curvature of different points on the object. But this method works in controlled conditions where the object to be grasped uniformly coloured and distinct from the background and this cannot be used reliably in various real world situations. A versatile closed loop method for reliable grasping of different types of objects at variety of locations does not exist yet.

A novel *active grasping* technique based on continuous object tracking has been proposed and developed in this thesis. It uses continuous object tracking as a visual feedback of the object's current position with respect to the robot. The concept of an end-effector manoeuvring based on the current location of the object to be grasped is called **Active grasping** as the robot is not passively executing a pre-generated motion, but actively moving towards the object. This method in the current state uses only a 1DoF revolute joint in the hip augmented by a 2DoF mobile base to grasp objects[4] from heights varying from a height of 150 cm to the ground. Any position in 3Dimensions with a single (fixed) orientation can be reached by the end-effector with the available 3 degrees of freedom. This concept can be later extended for additional degrees of freedom as well.

Functionally the *active grasping* manipulation process can be segregated into 3 main modules

- Approach
- Grasp
- Recovery Mechanism

The modules and their inter relationship are shown in Fig.5.2. The transition between these modules is brought about by a *state-machine* implementation.

A *Finite State Machine* built on top of a continuous controller is used to co-ordinate and accomplish both the *approach* and the *grasping* phase. There is also an additional emphasis to ensure a reliable object manipulation. The state machine and the modules involved are detailed in the following sections.

[4]Size of the Objects is limited by the capabilities of the gripper. Most of the personal use objects in everyday life were grasped.

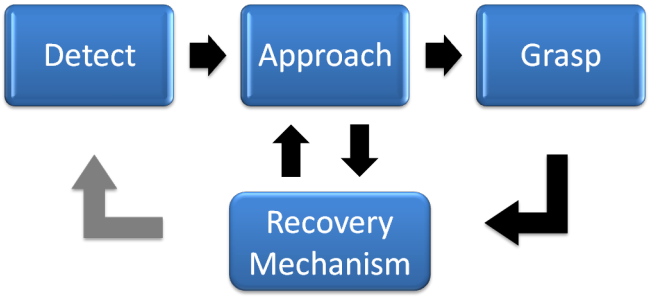

Figure 5.2: Functional Manipulation Modules

5.1 Grasping State Machine

The State machine mechanism developed for the grasping of any(previously unknown) object is shown in the Fig.5.3.

It can be seen that first step is the *object locater*. This is obtained from the modules

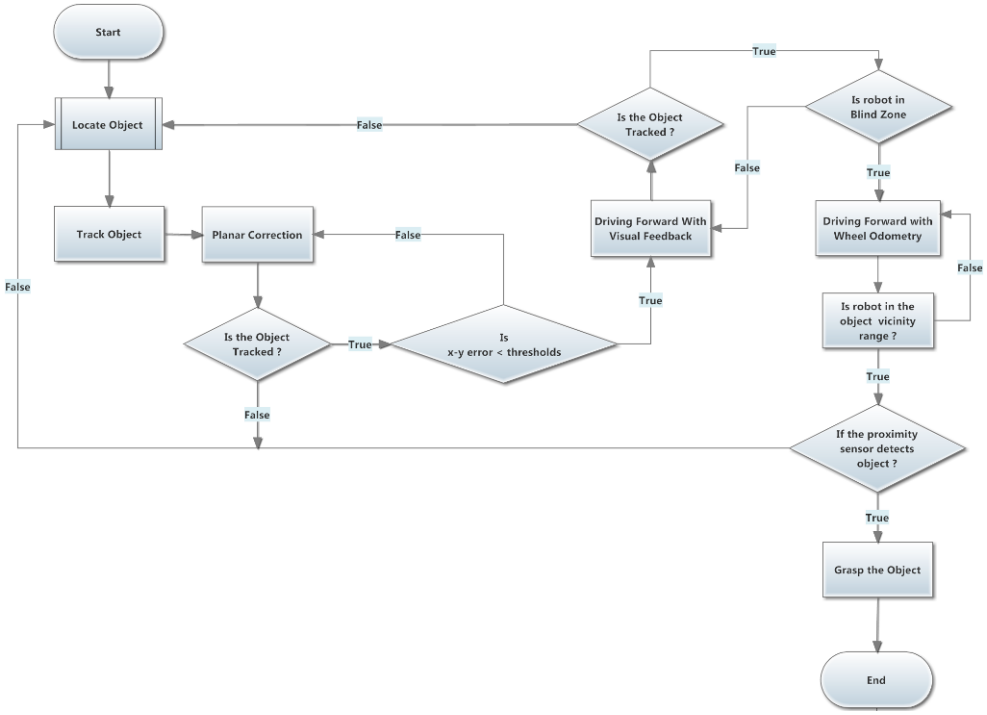

Figure 5.3: Grasping State Machine

explained in the previous chapters depending on the contextual requirements. For example if Cola can has to be picked up from a table, the objects on the table are located by the segmentation modules and the location of the required object is obtained from the recognition module. Once the object to be grasped is identified, it will be continuously tracked to provide the relative location feedback until it is grasped.

The grasping works principally on the visual feedback and hence the tracking needs to be robust and accurate. The tracking module is explained in the section *Model based tracking*. Once the object is being tracked, the relative position error between the robot hand and the required object is minimized until the object is in the reach of the gripper after which

the object is grasped. These constitute the *approach* and the *grasp* phase. There might be situations where the object tracking fails in midst of grasping process and this might lead to unpredictable reaction from the robot. In order to prevent this a certain safety measures have been embedded in the state machine for the robot to recover from loss of object position. This enhances the reliability of the entire grasping process. The operation of the robot in the *approach* and *grasp* phase and also the recovery mechanisms are further elucidated in the section *Approach - Grasp and Recovery*.

5.2 Model Based Object Tracking

Reliable tracking is the essential part of the entire grasping process. It can be seen from the flowchart in the Fig.5.3 that the entire state-machine is built over the tracking feedback. The concept of closed loop grasping was initially proposed and tested based on a simplistic *histogram matching* based tracker using the objects appearance in the HSV color space. But as the grasping algorithm increased in sophistication, the need for better tracking algorithms arose. The requirements of the tracker can be summarized as

Reasonable operating frequency As with any closed loop system, the frequency of the feedback is vital. Hence a module which can keep track of the object at atleast 10Hz is essential for the proper operation of the proposed algorithm.

Reliability of tracking The robots belief of the objects current relative location dictates its approach towards the object. While total loss of track of the object can be detected, tracking a wrong region which doesnot constitute the object leads to erratic behaviour from the robot. Hence the tracker which also provides the confidence of tracking the initially selected object is required.

Robustness While the tracking needs to be reliable to ensure that the robot is tracking the right object, it also has to be robust to occlusions, viewpoint variation and also has to regain the tracking of the object when the object gets back in the visual field after moved away.

Adaptive Behaviour As the robot might need to grasp previously unknown objects, it has to learn the models of the object to be tracked online. Also, since the appearance of the object being tracked varies continuously due to the robot approaching towards it, the tracker has to adapt to this new viewpoints by incrementally updating the model of the object. This is also essential in the previously explained bootstrapping process, to enhance the robots knowledge of the object being grasped.

From the various methods available in the literature, the TLD [29] was utilized for this purpose. **TLD**, which abbreviates for Tracking, Learning and Detection has the following salient features which satisfy the above stated tracker requirements.

- Realtime implementation with average 10 Hz operation on the computer used.
- Robustness to occlusions, illumination change, background clutter.
- Re-detection of the object after it is back in the visual field

These features are made possible by the underlying mechanism which is briefly described below.

Tracking Once the initial rectangle defining the boundaries of the object to be tracked is available, a *median-shift* tracker based on the Lucas Kanade method, updates the location of the object in the successive images through optical flow estimation.

Detection The optical flow based tracker cannot be effective in fast motions, sudden view point change, etc. Hence an object detector based on 2bit Binary Pattern features, which are learnt from the initial object boundary is used. The data from the detector and the meanshift tracker are fused to keep track of the object with increased confidence.

Learning It can be seen that the detector locates the object in the scene based on the initially learnt features. In order to continuously keep track of the object in varied viewpoints, the features of the object have to be updated at every frame. This has been implemented based on PN learning framework

With the fusion of these three mechanisms, the tracking requirements specified earlier can be satisfied. There have been certain circumstances identified, where this TLD tracker loses the object due to the lack of features to learn and track. Though the improved versions of this TLD tracker were proposed by Wang et al. [28], a real-time implementation compatible with the ROS software architecture is not yet available. Hence a simple *template matching* based method is used in conjunction to the TLD framework to ensure reliable realtime tracking of the objects. The entire tracking framework is based on 2D image data and the depth value corresponding to the tracked centroid is used to convert the tracked location in *pixels* to the actual distance in *metres* using the camera parameters explained and obtained in Eq.3.8.

Now the object can be tracked robustly in various environmental conditions, this information is used by the motion controller to approach and grasp the object. This process is explained in the next section.

5.3 Approach - Grasp and Recovery

The approach phase consists of manoeuvring the end-effector to a position where the object in consideration is within the reach of the gripper. Realizing this required end-effector position can be modelled as a control problem in 3 Dimensions($[x, y, z]$). The problem is simplified with the location of the object with respect to the end-effector being known.

If $[X_{obj}Y_{obj}Z_{obj}]$ is the current location of the object in the *gripper frame* of reference and $[X_{req}Y_{req}Z_{req}]$ is the location of the object in the same frame of reference, then a simple **Proportional** control law can be formulated as in Eq.5.3

$$V_x = -K_x(X_{obj} - X_{req}) \tag{5.1}$$
$$V_y = -K_y(Y_{obj} - Y_{req}) \tag{5.2}$$
$$V_z = -K_z(Z_{obj} - Z_{req}) \tag{5.3}$$

There are two transformations required for obtaining the object position in the *gripper frame*.

- Kinect To Neck transform
- Neck To Gripper transform

Kinect To Neck transform

The Kinect is a RGB-D camera which works on the reflection of structured light [24]. It has an IR projector which projects structured IR light on to the scene and there is an IR camera, which processes the reflection from the scene to generate a depth map. Because of this working principle, the depth measured by the Kinect is the distance perpendicular to the image plane of the Kinect cameras. All the points in the plane parallel to Kinect image plane have the same depth values. This is illustrated in Fig.5.4. Hence, the distance obtained from the Kinect can be projected on to a plane parallel to the robot's frontal plane to get the absolute distance from the Neck frame(where the Kinect is mounted). It can be seen that this is implied due to the rotation about the $x - axis$. Hence a transformation by rotating the obtained co-ordinates back to the horizontal(non-rotated) frame of reference. This is defined by the relation through the equations

$$Y_{neck} = y_{actual}\cos\theta_{neck} + z_{actual}\sin\theta_{neck}$$
$$Z_{neck} = z_{actual}\sin\theta_{neck} + z_{actual}\cos\theta_{neck}$$

where θ_{neck} is the angle of the neck tilt measured counter-clockwise from the horizontal. It can be seen that the iso-distant planes are uniquely determined by the *Pitch* angle of the Kinect with respect to the robot. The Neck is pitched in different angle depending on the circumstances and this angle is continuously read from the *Neck controller module*.

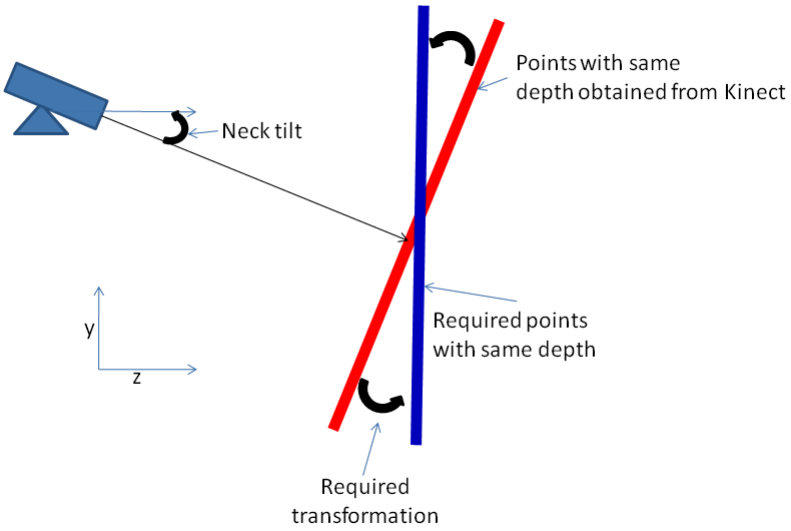

Figure 5.4: Kinect Transformations

Once the object position is known in the Neck frame, it is quite straight forward static transform to the Gripper frame. The relation is given in the Eq.5.4

$$\begin{bmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \end{bmatrix} = \begin{bmatrix} X_{neck} \\ Y_{neck} \\ Z_{neck} \end{bmatrix} + \begin{bmatrix} 0 \\ -0.15 \\ -0.05 \end{bmatrix} \tag{5.4}$$

Where $[X_{obj}Y_{obj}Z_{obj}]$ and $[X_{neck}Y_{neck}Z_{neck}]$ are the object position in the *Gripper* and the *Neck* frame respectively.

The transforms being available, the approach takes place in two stages.

- Planar$(x-y)$ Error Correction
- Depth(z) Error Correction

Planar Error Correction

The desired positions as in Eq.5.3 are set as $X_{req} = 0$ and $Y_{req} = 0$. Firstly any large error in X, Y directions are controlled and then the Depth Error is corrected.

Since the frequency of visual feedback from the tracking modules is not always consistent, a constant control frequency of 10 Hz is used to control the actuators. While the V_x controls the angular motion of the base, the V_y varies the height of the arm by controlling the revolute joint in the hip.

$$\begin{aligned} K_x &= 1.3 \\ K_y &= 0.8 \end{aligned}$$

These parameters were obtained based on response of the robot in different circumstances. If these gains are higher, the robot moves very fast and it can lose track of the object leading to grasping failure.
In case the gains are very low, the robot responds very slowly to the object and in circumstances when the object is externally moved, the robot can lose the object from its visual field again. Hence the above specified parameters were found to be optimal by testing on different objects and scenarios.

It was also observed from the implementation of this controller in practise that, there exists oscillation around the desired set point. This could be reasoned due to the presence of only

a *Proportional* controller. While an additional Derivative controller could be used to decrease this oscillation, it would lead to a longer *settling time* which is not very much desired for a realtime operation. This can also be easily affected by noise in the visual feedback. Additionally a second order controller can lead to instability with the presence of a PID controller in the 3Mxel motor controller. Hence a simpler and effective solution of using a region of desired position instead of a final desired position. Hence the Eq.5.3 is modified to Eq.5.6

$$V_x = \begin{cases} -K_x(X_{obj} - X_{req}) & \text{if } |X_{obj} - X_{req}| > X_{threshold} \\ 0 & \text{if } |X_{obj} - X_{req}| \leq X_{threshold} \end{cases} \quad (5.5)$$

$$V_y = \begin{cases} -K_y(Y_{obj} - Y_{req}) & \text{if } |Y_{obj} - Y_{req}| > Y_{threshold} \\ 0 & \text{if } |Y_{obj} - Y_{req}| \leq Y_{threshold} \end{cases} \quad (5.6)$$

Where the parameters

$$\begin{aligned} X_{threshold} &= 5mm \\ Y_{threshold} &= 20mm \end{aligned}$$

are used. The horizontal alignment of the gripper with the object has to be more precise than the vertical alignment. This is because even if the gripper slightly misaligned horizontally, the object will be off centred within the gripper and there is a greater probability for the failure of final gripping. This is reflected in the tolerance in $X_{threshold}$. The tolerance in $Y_{threshold}$ can be higher as the 3 gripper phalanges allow successful grasping even slightly above or below the centroid of the object.

An ellipse centred at X_{req}, Y_{req} and its principle axes dictated by the *threshold* parameters is the *Tolerance Zone*. The system is actuated until the positional error trajectory enters this zone. The phase plot showing variation in $x - y$ errors for objects at different heights can be seen in the Fig.5.5.

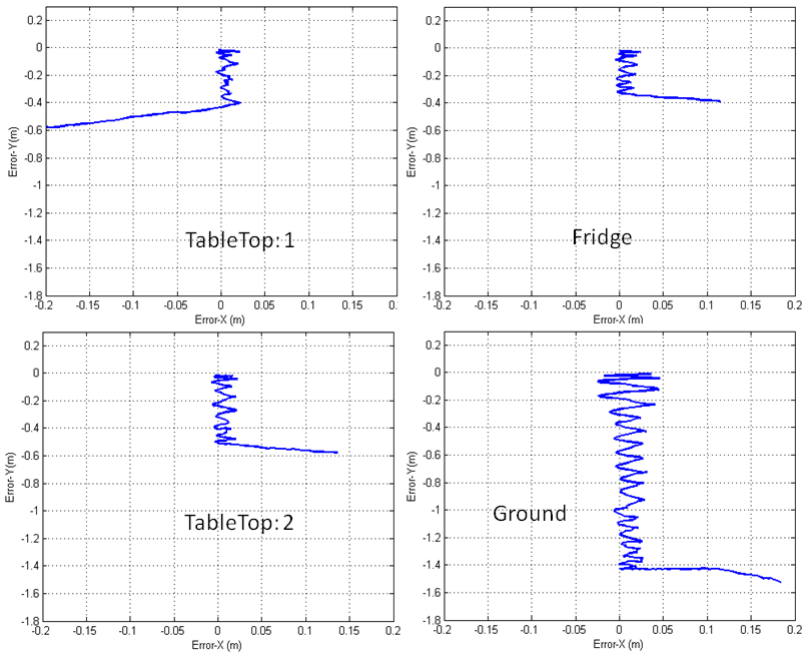

Figure 5.5: Phase Plot of the Error

Depth Error Correction

The $x - y$ correction ensures that any large planar error is within the threshold range and ensures the robot(gripper) is coplanar with the object plane. In ideal situations the robot has to move straight forward to grasp the object. Instead of blindly moving forward until

the object is grasped, additional intelligence has been incorporated. This takes place in two states.

1. Driving Forward with Visual Feedback
2. Driving Forward with Wheel Odometry

Driving Forward with Visual Feedback

The lower range of the depth map obtained by the Kinect is limited by its working principle [24] to be **45 cm**. Within this distance region the depth information from the Kinect is not available. This region of depth in front of the Kinect is termed as its **Blind Zone**. Due to this, the robot initially drives forward decreasing the distance error based on the control law on z direction from Eq.5.3 which is

$$V_z = -K_z(Z_{obj} - Z_{req})$$

until it reaches the Blind zone. A value of $K_z = 1$ was found to be adequate for the direct approach of the object. Once the robot's presence in the blind zone is detected in the tracking feedback, the robot moves forward relying solely on its *Proprioception*. Proprioception is the knowledge of the robots current configuration based on the internal sensory information. For example, the position of the end effector can be predicted if the joint angles are known. This can be seen as analogous to the way humans can reach small distances with their arms in the dark. In this case, the distance moved by the robot is approximated by using the velocity commands issued to the mobile base.

Driving Forward with Wheel Odometry

The last known distance from the object($Distance_{blind}$) before the blind zone is the distance to be moved front before the robot is just in front of the object. The object is grasped when the proximity sensor(refer sharp sensor again) detects the presence of an object at a distance less than $Distance_{grasp}$ from the gripper. This value was determined to be

$$Distance_{grasp} = 6cm$$

A higher value causes the gripper to be closed, even when the object is not within it. When a lower value is used, the robot can topple the object while trying to be very close to it. Hence the value of $6cm$ was found to be optimal for a successful grasp for almost all the everyday used objects.

In order to avoid any stray sensor noise influencing the grasping, an additional check of wheel odometry(which is the robot's internal measure of its motion) is used. Since the velocity commands to the base is known, the distance$[S_{forward}(t)]$ moved forward at any time instant T can be determined by the simple integral as in Eq.5.7

$$S_{forward}(T) = \int_0^T V_z(t)\,\mathrm{d}t \tag{5.7}$$

Based on this, the robot can grasp the objects based on the proximity sensor information, only if the distance moved $S_{forward}(t)$ is within a threshold distance($Thresh_{odom}$) around the $Distance_{blind}$. So the final grasp based on the sensor is activated by the state $Grasp_{state}$ which is governed by the Eq.5.8

$$Grasp_{state} = \begin{cases} True & \text{if } |S_{forward} - Distance_{blind}| \leq Thresh_{odom} \\ False & \text{otherwise} \end{cases} \tag{5.8}$$

The $Thresh_{odom}$ is used instead of using the exact absolute $Distance_{blind}$ in order to compensate for the inaccuracy in

- Kinect Depth Information
- Wheel Odometry Information

The value of

$$Thresh_{odom} = 5cm$$

was used to obtain reliable grasping in different indoor environmental conditions. Once these conditions are satisfied and the object is grasped, the robot returns to the configuration (default neck and hip joint positions). Then the overall object manipulation module returns a *success* response.

The overall object grasping process being closed loop, this technique can be extended to the grasping of dynamically moved objects. The robot keeps track of the object's current location even if it is moved intentionally by a human. Hence the above described process can be directly used in these situations to approach and grasp a moving object as well, leading to an *interactive grasping* process.

Validation

This grasping strategy was evaluated by grasping 10 different objects from different locations. The locations used were

- Table Top
- Floor
- Frige Shelf

This was also evaluated on plain and multi-coloured background. The objects were also grouped into simple and complex based on the ease of grasping. This grouping is shown in the Fig.5.6. The influence of presence of other objects near the object to be grasped was also

Figure 5.6: Simple and Complex Objects

studied. The plots listed below show the invariance of this grasping technique's performance in these different conditions. These plots show the convergence of $x-y$ errors until the robot is in the blind zone of Kinect after which the robot approaches the object straightforward.

- The plot in Fig.5.7 shows the error convergence for simple and complex objects at different locations when only the object to be grasped is present
- The plot in Fig.5.8 shows the convergence for single and multiple object scenario proving that presence of other objects does not affect this grasping process
- The variation of the error during an interactive grasping process is showed in the Fig.5.11
- The convergence of the error in height while grasping objects from different heights can be clearly seen in the Fig.5.9

A sequence of robot grasping an object is shown in the Fig.5.10 and the videos of the entire grasping process can be found in the website[5].

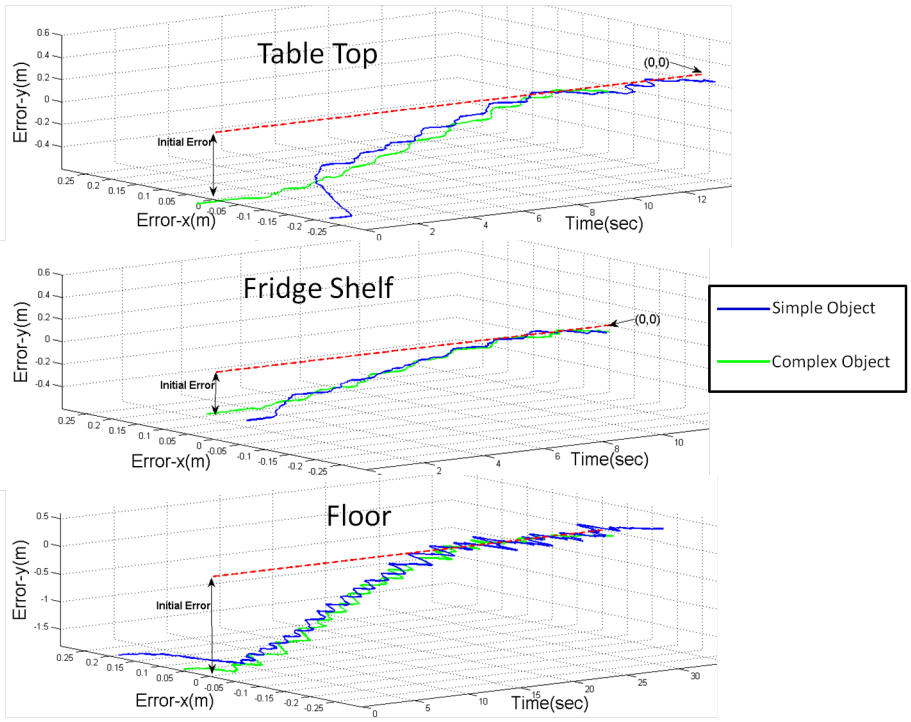

Figure 5.7: Grasping from different locations

Table 5.1: Overall Grasping Time(sec) at Different Locations

TableTop	Floor	Fridge
24.51	51.65	21.08

The average grasping time was influenced only by the location of the objects but not on the complexity of the object. This grasping time for different locations evaluated are tabulated in the table: 5.1.

The reliability of the grasping process is evaluated and the precision is tabulated in table: 5.2. It can be seen that the grasping can fail when the distance between the gripper and the object is larger. This can occur due to the tracker or due to the gripper. It is possible that when the grasping process is initiated again on the same object, the grasping will succeed. Hence, it is essential for an intelligent robot to identify that the grasping had failed and have to respond accordingly. These are the *Recovery Mechanisms* and is explained in the next section.

Table 5.2: Precision of Grasping Simple and Complex Objects from Different Locations(%)

	TableTop	Fridge	Floor
Simple Object	100	80	75
Complex Object	95	80	70

5.4 Recovery Mechanisms

The entire manipulation process described above function assuming the object position is continuously available from the *tracking* module. The manipulation process can go wrong in situations where the *tracker* module temporarily or permanently loses the location of the object. In order to avoid undesirable behaviour from the robot, certain measures are taken for the robot to recover from these situations. This recovery action is two folded.

[5]http://www.delftrobotics.nl/galleries

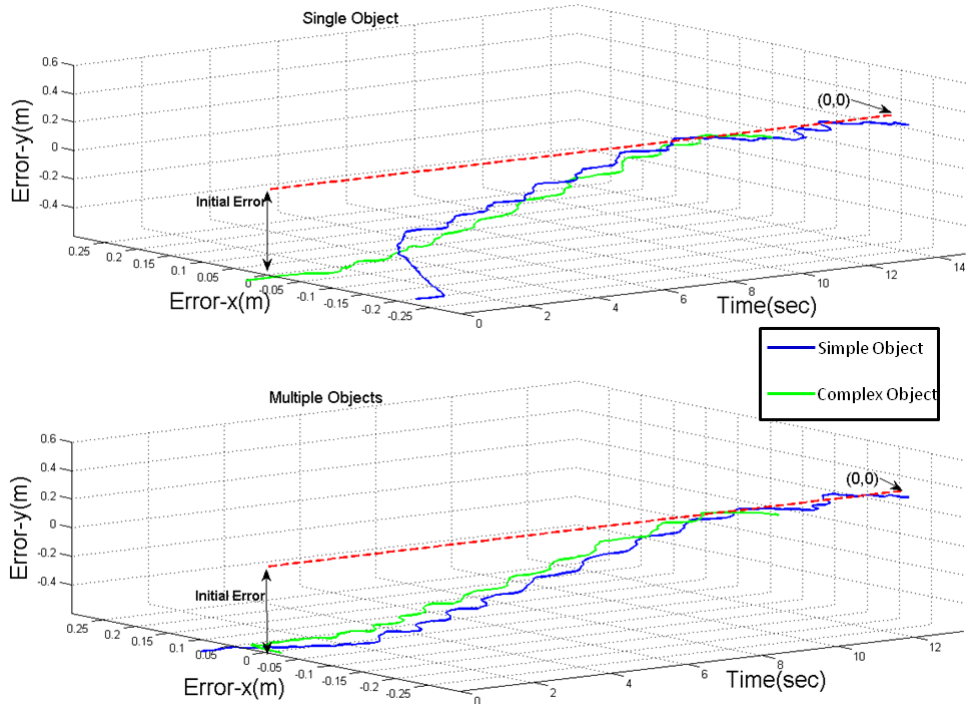

Figure 5.8: Single and multiple object scenario

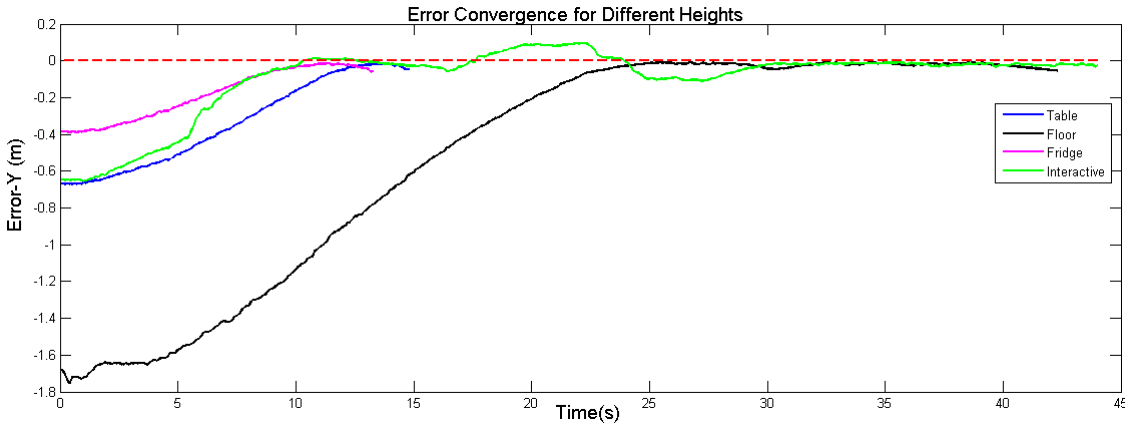

Figure 5.9: Height(y) convergence at different locations

- Recovery in Approach phase
- Recovery in Grasp phase

Recovery in Approach phase

In approach phase the controller works purely on the basis of the visual feedback. There are instances where the tracker temporarily loses the object due to sudden view point change. In this situation the robot attempts to regain the tracking by retracing to the last position where the object was still tracked.

In most of the situations the tracker regained the object and the controller functions normally again. In case when the object location cannot be regained the robot acquires the initial configuration at which grasping module was activated. The state machine also switches to the *Object locater* state where the object to be grasped is located again and the tracker starts by learning the features afresh.

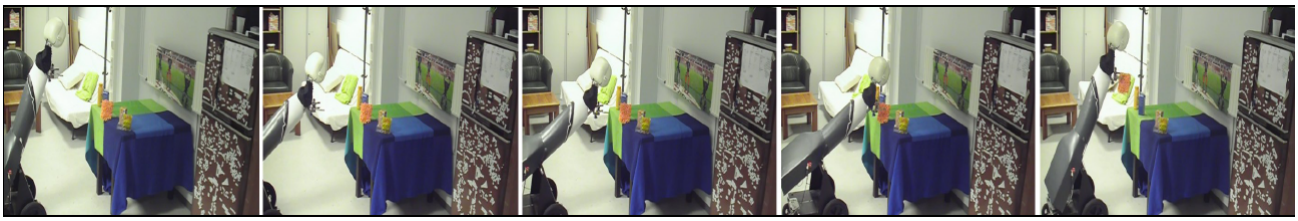

Figure 5.10: Sequence of Grasping from Table

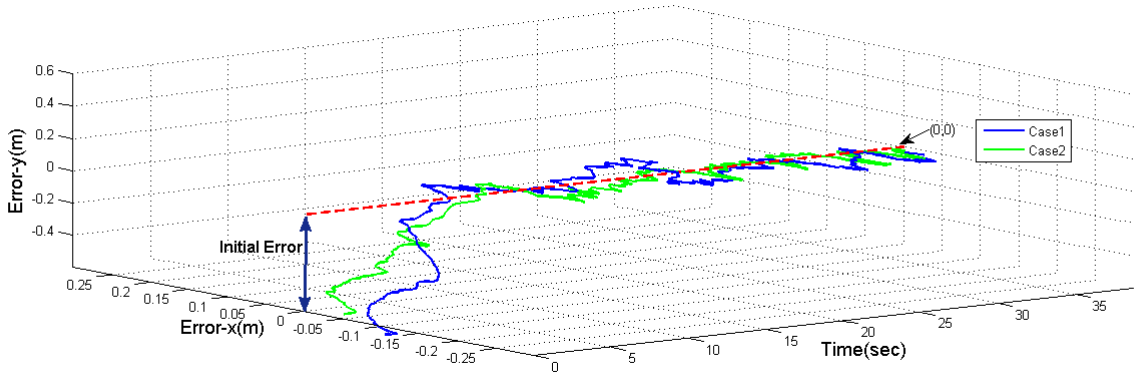

Figure 5.11: Error Convergence in Interactive Grasping

Recovery in Grasp phase

When the approach phase has been successful and the robot is in the depth correction phase, it has been explained that the grasping is accomplished only when the *proximity sensor* detects the presence of objects when the $S_{forward}$ satisfies the condition imposed by the Eq.5.8. When the object is not detected by the *proximity sensor* even while the robot is in the vicinity of the object described by Eq.5.8 the only possible reason is that, the object is not in the location anymore. Hence the robot switches back to the *Object locater* state and the whole process is repeated again.

It is restricted to a maximum of 3 attempts to start from the initial(*Object locater*) state. If the object is failed to be grasped within three times, the robot stops to try and assumes that the object is not graspable. Then the overall manipulation module returns a *failure* response.

It can be seen from the Fig.5.5, Fig.5.7 and Fig.5.11 that this grasping technique is generic and simplistic. It can be reliable in many scenarios like simple objects, complex objects, different locations and also externally movable objects. These features of the grasping technique make the grasping process smooth, continuous and a naturally interactive process also leading to better human-robot interaction.

Chapter 6

Conclusion

The problem of entirely bootstrapping the knowledge of a robot's environment is very elaborate. One small aspect of understanding the objects in the environment has been approached in this thesis. The process that contribute towards this incremental development of the robots understanding of its surroundings in the context of objects present have been identified. The framework of robot improving this knowledge is extended to incorporate increased sensory motor integration.

The visual field of the scene is obtained as an RGB image augmented with the spatial information obtained from the Kinect. This information was processed to locate the objects in 3 dimensions. This was projected back to the object boundaries in the RGB image and a better quality image of the same object was obtained by transforming this boundaries into the image from a calibrated high resolution camera.

The appearance properties of the objects are described as feature vectors and this is used to recognizing them. A novel generic closed-loop grasping technique which operates independently of the location or the properties of the object has been proposed, implemented and validated. This is based on the feedback from the tracking module that continuously keeps track of the object. This process can be used to grasp a recognized object or instead used to learn an unknown object by the grasping process.

The proposed grasping process simultaneously learns the object's appearance while the object is being grasped. This can be extended to incorporate robotic arms with more degrees of freedom which allows manipulation with different orientations.

Bibliography

[1] http://classics.mit.edu/aristotle/politics.1.one.html.

[2] https://alliance.seas.upenn.edu/ meam620/wiki/index.php?n=roslab.ipcbridge.

[3] T. Asfour D. Kraft S. Knoop R. Dillmann A. Kargov C. Pylatiuk S. Schulz A. Morales, P. Azad. An anthropomorphic grasping approach for an assistant humanoid robot. *37th International Symposium on Robotics*, pages 149–152, 2006.

[4] Mukunda Bharatheesha, Maja Rudinac, Aswin Chandarr, Machiel Bruinink Floris Gaisser, Susana Pons Rueda, Berend Kupers, Sep Driessen, Xin Wang, Martijn Wisse, and Pieter Jonker. Delft robotics robocup@home 2012, team description paper.

[5] Jean-Yves Bouguet. http://www.vision.caltech.edu/bouguetj/calibdoc/index.html.

[6] Berk Calli, Martijn Wisse, and Pieter Jonker. Grasping of unknown objects via curvature maximization using active vision. *Intelligent Robots and Systems (IROS),*, pages 995–1001, 2011.

[7] Marco Ceccarelli, editor. *International Symposium on History of Machines and Mechanisms*. Kluwer Academic Publishers, 2004.

[8] D.G.Lowe. Distinctive image features from scale invariant key points. *IJCV*, 60(2):91–110, 2004.

[9] T. Gevers E. A. van de Sande and C. G. M. Snoek. Evaluation of color descriptors for object and scene recognition. *IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska, USA,*, 2008.

[10] Rosheim; Mark Elling. *Leonardo's Lost Robots.* Springer, 2006.

[11] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[12] M. Edwards G. M. Bone, A. Lambert. Automated modelling and robotic grasping of unknown three-dimensional objects,. *IEEE International Conference on Robotics and Automation, Pasadena, CA, USA.*, pages 292–298, 2008.

[13] Hou and L. Zhang. Saliency detection: A spectral residual approach. *CVPR*, pages 1–8, 2007.

[14] S. Jafari, R. Jarvis, and T. Sivahumaran. Relative visual servoing. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 2, pages 880 – 885 vol.2, dec. 2004.

[15] Gert. A. Kragten. *Underactuated Hands: Fundamentals, Performance Analysis and Design*. PhD thesis, Delft University of Technology, 2011.

[16] Jr. Kuffner, J.J. and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 995 –1001 vol.2, 2000.

[17] K. Lai, Liefeng Bo, Xiaofeng Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817 –1824, may 2011.

[18] M. Vincze M. Richtsfeld. Grasping of unknown objects from a table top,. *ECCV Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments.Marseille, France,*, 2008.

[19] M. Rudinac and P. P. Jonker. A fast and robust descriptor for multipleview object recognition. *ICARCV*, pages 2166–2171, 2010.

[20] M. Rudinac and P. P. Jonker. Saliency detection and object localization in indoor environments. *ICPR , IEEE*, pages 404–407, 2010.

[21] Maja Rudinac, Gert Kootstra, Danica Kragic, and Pieter Jonker. Active learning and recognition of objects in the unstructured environment. In *IEEE International Conference on Intelligent Robots and Systems IROS*, 2012.

[22] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John 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, October 2010.

[23] Noel Sharkey. http://www.shef.ac.uk/marketing/eview/articles58/robot.

[24] Zalevsky Zeev Shpunt Alexander. Three-dimensional sensing using speckle patterns, April 2009.

[25] Ioan Alexandru Sucan and Lydia E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *WAFR'08*, pages 449–464, 2008.

[26] M. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology,*, 12:97–136, 1980.

[27] Vitruvius. *The Ten Books on Architecture.*

[28] Xin Wang, Maja Rudinac, and Pieter P. Jonker. Robust online segmentation of unknown objects for mobile robots. In *VISAPP (1)*, pages 365–374, 2012.

[29] Krystian Mikolajczyk Zdenek Kalal, Jiri Matas. Online learning of robust object detectors during unstable tracking. *3rd On-line Learning for Computer Vision Workshop,Kyoto, Japan, IEEE CS.*, 2009.

[30] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *Computer Vision, IEEE International Conference on*, 1:666, 1999.