Tampere University

Toan Le

VISUAL PERCEPTION WITH UR5 ARM TOWARDS SEMANTIC MANIPULATION

Faculty of Engineering and Natural Sciences (ENS)
Master Thesis
April 2020

ABSTRACT

Toan Le: Visual perception with UR5 arm towards semantic manipulation
Master Thesis
Tampere University
Automation Engineering - Robotics
April 2020

Nowadays, the most popular impression about robotics is the factory assembly line of robotic arms working together. Although robotic manipulation has substantially grown in the last decade, creating robots capable of directly interacting with the surrounding is still a desire, where the robots could effectively and autonomously function and safely interact with the human. To achieve this, the robot must have the ability to understand its surroundings. From my perspective, the ontology is a good choice to be a robot knowledge base of the environment. With a knowledge base as the central server, the problem is how to make the system be flexible for expanding purposes. This thesis presents an approach to tackle this problem by combining the ontology, web server, and ROS environment, which is used as a control system of robot and sensors. The target outcome is that when the system analyzes a current random scene, it can effectively detect the differences between the current scene and the target scene, i.e if there is an object that is not in the target scene, the robot needs to remove it from the current scene. As a result, the proposed system generates the equivalent actions and gives commands for the robot to act accordingly. Thanks to the above process, the robot successfully accomplishes the target tasks from random scenes.

Keywords: Robotics manipulation, Visual perception, UR5, Semantic manipulation, Human-robot Interaction

PREFACE

This thesis work was conducted in the Cognitive Robotics research group at Tampere University. In the six months in Tampere and more three months in Helsinki, I had a chance to complete my thesis in the supporting environment around intelligent and kind people. I would like to give my sincere thanks to my instructor PhD Candidate Alexandre Angleraud and my supervisor Assistant Professor Roel Pieters for their patience and supporting me in this work. I want to thank Tampere University people for carefully helps and the good memory since the first time I came to study at the Hervanta campus.

Last but not least, I want to give my heartfelt thanks to my Mom, my Dad for their belief in me. Thanks to Hong for always being by my side, I want to say from the deepest of my heart that the world would be a mono-color picture without you. Thanks to my best friends Tran and Ngoc, I want to thank you guys for every unstoppable laughing moment we have since the newbie party.

Helsinki, 29th April 2020

Toan Le

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
CMY	Cyan Magenta Yellow color channel
HRI	Human-robot Interaction
HSV	Hue Saturation Value color channel
OWL	Web Ontology Language
PCL	Point Cloud Library
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RGB	Red Green Blue color channel
RMSE	Root Mean Square Error
ROS	Robot Operating System
SOMs	Semantic Object Maps
UI	User Interface
W3C	World Wide Web Consortium

1 INTRODUCTION

Based on function and structure, robotics has two main majors namely robotic manipulator and mobile robot. Last decade has seen a substantial growth in both majors where the robot now could model the environment as 3D map for localization, or use camera to recognize different objects for manipulation, or simultaneously do both task as in Zeng et al work [54]. Thanks to the development of vision technology and computational power, it is now possible for a robot to accomplish difficult tasks with visual guiding [56]. However, creating a robot capable of directly interacting with environment and safely working in human-living environment is still an aspiration. Before accomplishing that desire, there are various obstacles to tackle. Firstly, the robot needs to have information about the surrounding. The robot also needs the ability to process the data to information that it could understand and store the information for later using.

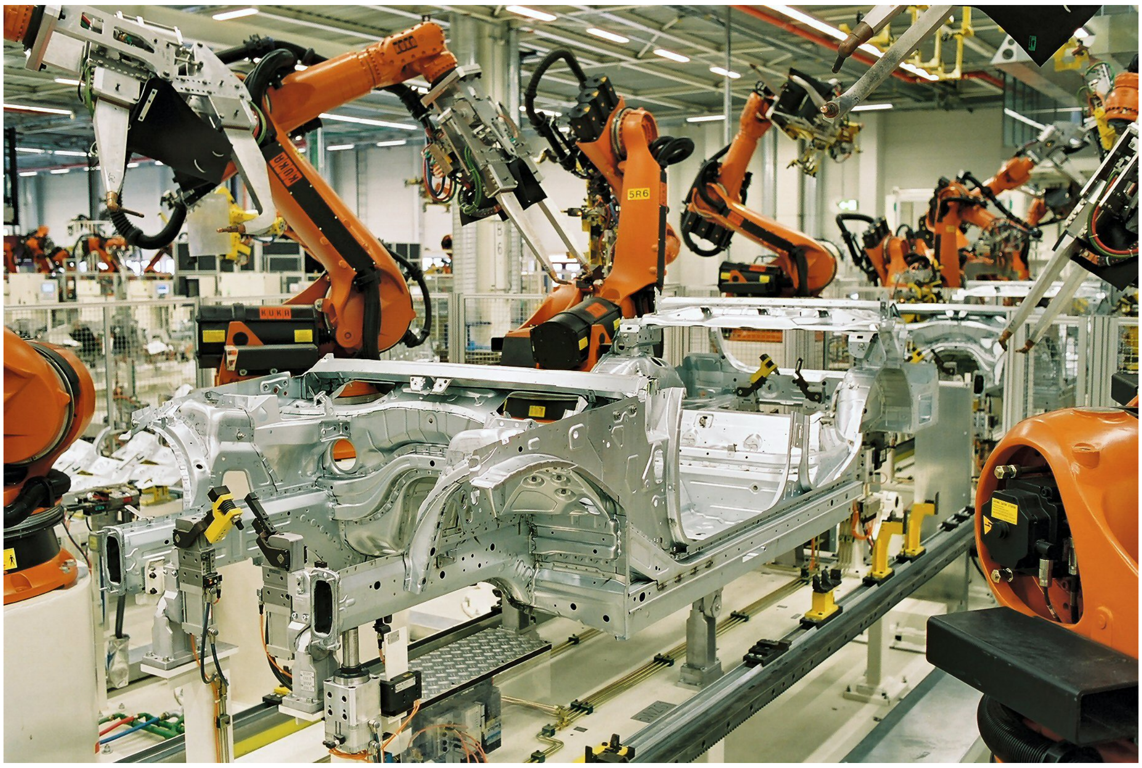


Figure 1.1. *Robotic manipulators in car industry [8]*

In this chapter, the motivation and justification for problem statement are presented below. Furthermore, this chapter introduces the key questions, that the thesis aims to solve, the limitation and objectives considered in thesis implementation duration.

1.1 Motivation

Robotic manipulators or robot arms play the main role in robotics development. Nowadays, factories with multiple robotic arms working together is a popular image but the robotic arms are attached in same position doing same actions, what if those arms could move and work in random scenarios instead of well-defined environments. Indeed, human's desire of creating an artificial assistant working properly in the human-living environment has led to the race to develop robot for service purposes, which is the combination of a robotic arm attached to a mobile base [23]. These kind of robots have great potential in nursing, providing care for elderly [29, 53], which were seen in movies for long but has not been brought in service until recently. One of the most challenging requirements for the robot to work safely in human being environment is that it could perceive the surrounding. The robot will act based on that understanding and has the ability to expand its knowledge to adapt changes. This concept is termed as Semantic Manipulation.

Researches in semantic manipulation are extensive and diverse. To tackle the understanding problem of robot, a majority of the works in semantic manipulation focus on object detection and grasping optimization [2, 7, 55], while some did aim to improve the interaction between human and manipulation system [9, 48]. Thanks to the development of computational power, there is significant leap of object detection field in recent decades where the robot not only recognize the object in cluster scene but also estimate the location of desire object in real world [3]. Equally, the grasping optimization has been developing for past several decades with aiming at solving the problem of incomplete knowledge of environment. From the achievement of both fields, the goals of Semantic Manipulation are feasible to achieve but it requires the manipulation system being programmed by highly trained professionals. It would be difficult to actually apply the system in human-living environment due to limitation of user experience and lacking of the ability of adaptation. In this concept, user experience is the human-robot interaction (HRI), which should provide the user that has no knowledge in the field of robotics, a capability of programming and using the robot comfortably. On the other hand, the ability of adaption allows the system to be modulized for potential of expansion. Over the past years, many research works have been conducted in improving interaction between human and robot, which is usually demonstrated by two popular approaches as teaching panel and software API. The drawback of these approaches is that it requires user to be in close proximity to robots while there is a trend to enable controlling them from distance. Besides, real world is diverse and the same task can be performed in different environment. It is a great burden to design or make modification for each set of environments. This burden can be eliminated by enabling the robot's self adjustment or system modulization.

For the above reasons, this thesis presents an orchestration framework for a robotic manipulator. This framework enables a robotic arm to perceive and interact with the real world in a more robust manner by utilizing the knowledge base sharing among the robot itself and other sensing elements. The operator must has ability to modify manipulation task and control the robot through web interface, which provides ability of remote controlling.

1.2 Research Questions

This work addresses the mentioned problems and by solving those, it brings the opportunity to enhance the flexibility and expand the user experiment of semantic manipulation system. To achieve that purpose, this thesis focuses on finding the answers for these following questions:

- Is it possible to modulize the proposed framework?
- How can the user interact with the system from distance?

1.3 Objectives and Research Scope

In order to successfully achieve and implement the goal of this thesis, the objectives should be defined.

In this work, we propose the framework of an application which allows an unrelated-robotics background user to control intuitively a robotic manipulator to handle a simple pick and place task. The user has the ability to update the knowledge base of the robot and controls it to do the sequential primitive actions for accomplishing the tasks, which is defined in the knowledge base. To do that, there must be a pipeline between Robot Operation System (ROS) and user interface. The proposed framework must have ability to substitute its modules.

Due to the limitation in time and equipment, this work will be developed mostly in simulation. To demonstrate the reproducibility, more repetitions and experiments on real equipment are needed.

1.4 Document Structure

This document is organized as follows: Chapter 2 gives an overview about the background theories, the ontology and reason why ontology is used in this work instead of other data storage. Furthermore, this chapter reviews the basic concepts of image processing, which is used as the engine of visual manipulation in this work. The brief review of semantic web as user interface is also addressed in this chapter. Chapter 3 explains the research methodology, from creating simulation environment to developing the ontology in order to tackle the addressed issues. The experiment and building simulation environment is discussed in Chapter 4 while Chapter 5 provides a summary of this work and possible future plan to improve what is archived.

2 THEORETICAL BACKGROUND

2.1 Robot Vision

Robot vision is a related branch from the field of computer vision, besides machine vision, where the host computer not only processing the environment data collected from cameras or sensors but also analysing the data and using it to control the client robot. In this section, the theory base is divided into two parts to clarify its role in this thesis topic. First part discusses image processing and introduce some basic algorithms that are used in processing task. Image processing task aims at extracting features of surrounding environment and converting them into digital data that can be handled by robot or computer. Thanks to the richness of information an image can provide, the robot has ability to locate itself in current surroundings and make necessary adjustments to perform the current task. The second part talks about how to use results from the previous step to control the robot and this task is termed as *visual servoing*. Besides, based on the sensor position, there are two schemes to control the robot: *position-based visual servoing* and *image-based visual servoing*. From that reason, the data that describes the outside world, has to pass through two processes sequentially, which was also described in the work of Berthold in 1986 [16]. Firstly, an image is captured and its data is collected and converted into digital form in the host computer. The host computer then uses these data to generate actions and command the robot to do it. The flow of data basically is the conversion from information to physical actions and the flow is presented as picture below.

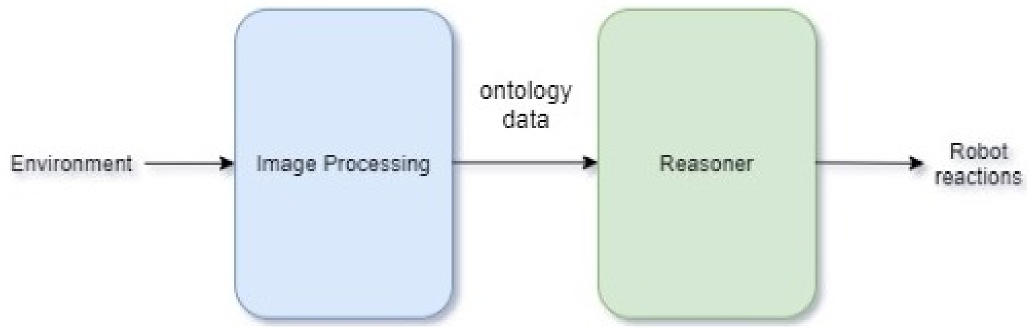


Figure 2.1. *Vision processing*

2.1.1 Image Processing

Image processing is the popular term nowadays due to the huge expansion of technology market where mobile smart device is an obvious example. One of the most important product features that every world-brand technology corporations want to advertise to customers is how the good picture their device can provide. Hence, the question is what image processing is and why it has strong relation to many technical areas.

Getting a meaningful image is desired by many as it is related to how human perceive information from surrounding environment. Human perception is a fusion of multi-sensors system where vision, our eyes, is the most important. The way people see the world can be presented through machine eyes, in this circumstance the environment data can be illustrated as images. However, there are differences between the way a human sees a picture with that of a robot sees a picture. Human can detect what is in the picture, differences of colors and even the occasion of the picture while robot collects digital information and mainly just using few potential of that information because of lacking technology in the past.

At this instant, the technology with high power GPU or even the CPU is also power to analyse and process the information that a machine can get from a picture. With enough data, the machine could predict or perceive the scenario of the picture somehow similar as the way human can. In this case, a robot has ability to react to what is happening in real world because it can somehow "understanding" the current scenario, as already mentioned in Chapter 1.

Back to the topic of this section, image processing is a task of analysing input image for information collecting. This work is performed by seeing input picture under different perspectives to give an output, which could be a new image with desired information included or a conclusion for a query from user or for storing information in ontology. Simple flow of image processing is showed below.

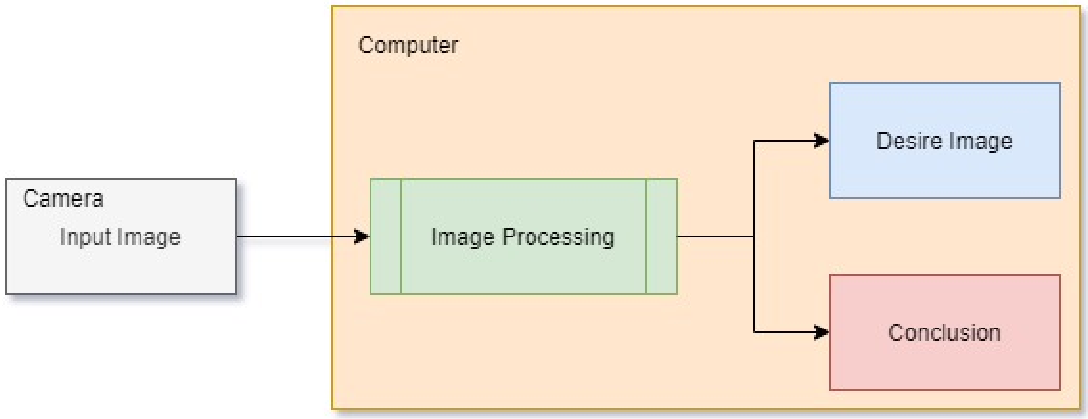


Figure 2.2. *Image processing*

The image is set of points or pixels. These points and pixels represent light intensity or the feature of specific coordinates in space, which is the result of the combination of multiple sources. Therefore, processing an image is similar as solving an equation with multi-variables. In detail, image processing task, which is the green block of Figure 2.2, is somewhat similar to peeling an onion, layer by layer, as illustrated in Figure 2.3.

The pre-processing block prepares data for next step. For example, it down scales the image or changes it color channels or simple as converting input data type to another that is more suitable for processor to work with. Next, the features extraction block analyses the data from previous step. Many types and algorithms could be implemented in this block depending on what is considered as features of interest for user and a more detailed discussion on this topic can be found in the next chapter. Shortly, under the scope of this thesis work, the features extraction block detects the color of the desired object and their relative positions of them with reference to the camera position. Finally, the post-processing is where the collected features are organized for storing or transferred to use in other task, which is used to query the spatial information in this thesis.

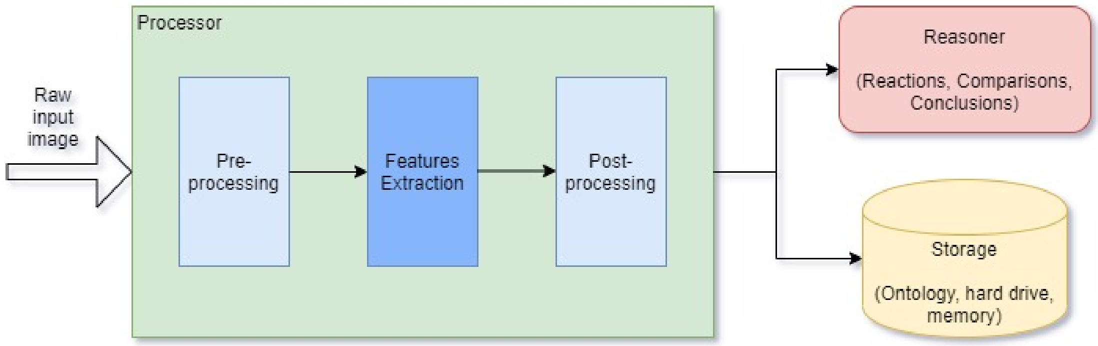


Figure 2.3. *Primitive steps of an image processing system.*

Basic Concepts

- **Images and pixels:** As a common definition, pixels are fundamental units of an image. A pixel displays the light intensity of a location of an image. Image, vice versa, is a set of pixels. Human observes an image as a whole while robot or computer would retrieve information at a deeper and more detailed level by analysing each pixel of that image.
- **Grayscale and color channels:** These parameters perform as possible numeric values of pixels in an image.

Color Channels

Human eye is a special "sensor" with unique design that allows vision - the ability to visually perceive the surroundings. Although we could only name dozens of colors, the eye can distinguish thousand of colors, if not millions. For example, when putting a blue pen in a light room, you see it blue but when taking it outdoor in natural lighting condition, you could see it a little darker or brighter. Basically, the pen is made with one intended color - blue - but the perceived color depends on environment conditions such as light setting. In addition, color is the combination of three properties which are Hue, Saturation, Intensity.

In image processing and graphics, a color can be presented as a numerical parameter, given that the color can be created by combining primary colors at a certain mixing ratio. Color model is a structure that helps describing a color by defining its location in 3D or 4D coordinates or the mixing ratio of 3 or 4 primary colors to create it, where each primary color is a channel or a dimension in the model. For example, RGB model is the cube where its parameters as width, length and height play roles of Red, Green and Blue accordingly.

The main purpose of using color channels is using numeric parameters under mathematical perspectives to portrait images. Subsequently, the robot or computer could comprehend the image under number or math presentation. Besides, extending ability to feel the colors of the robot is another benefit from color channels. As a result, the color model modifying or color filtering is the popular and uncomplicated approach to process an image. Couple of models will be suggested below.

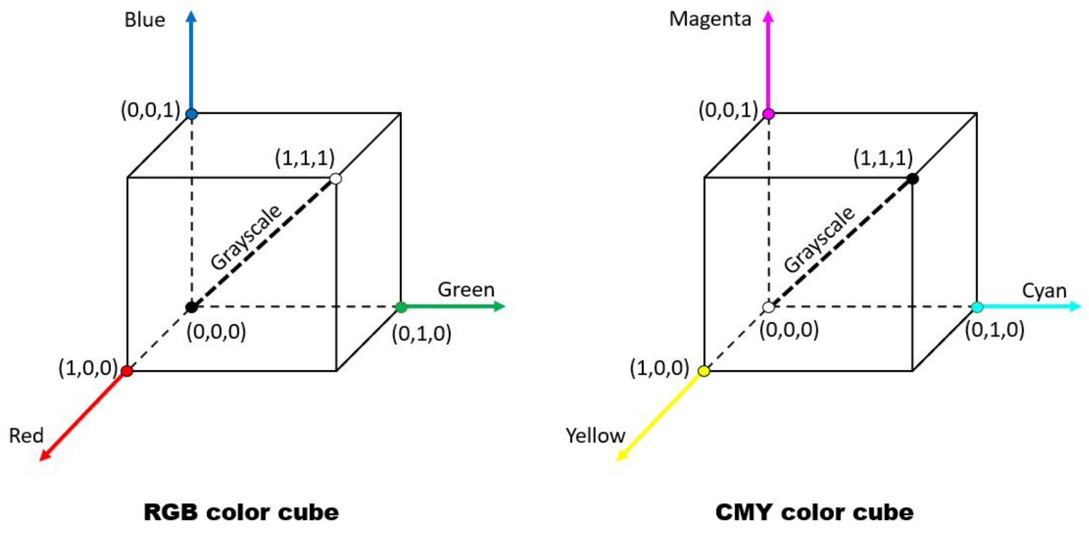


Figure 2.4. *RGB - CMY color space cube.*

- **RGB model** Red - Green - Blue model is the most popular in-used color model. There are three primitive colors in this model which are the main axes of the color coordinates. To perform other color, the primitive units will be adjusted to get close to the target color. In order to get the shade of target color, the main diagonal of RGB cube is curved. This main diagonal is start from the coordinates origin (0,0,0), where is totally black which means no color, to location of (1,1,1), where the color is fully white due to it is the converging of maximum primitive values. For visualization, the RGB cube is drawn in Figure 2.4.

- **CMY model**

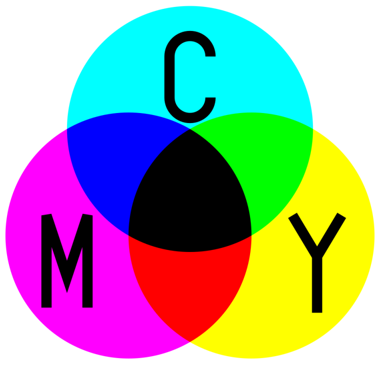


Figure 2.5. *CMY color wheel [17].*

Cyan, Magenta, Yellow are the complements of Red, Green, Blue, respectively, and Cyan, Magenta, Yellow also are used to filter out three primitive colors mentioned above from white light. That is the reason why CMY is known as substractive color model of RGB model. The collection of performing colors in CMY model is also applied as color cube in Figure 2.4. The difference between RGB cube and CMY cube is the origin of grayscale line where it is (0,0,0) value but displaying white color instead of black. The relations between CMY and RGB colors is mostly performed as color wheel in Figure 2.5.

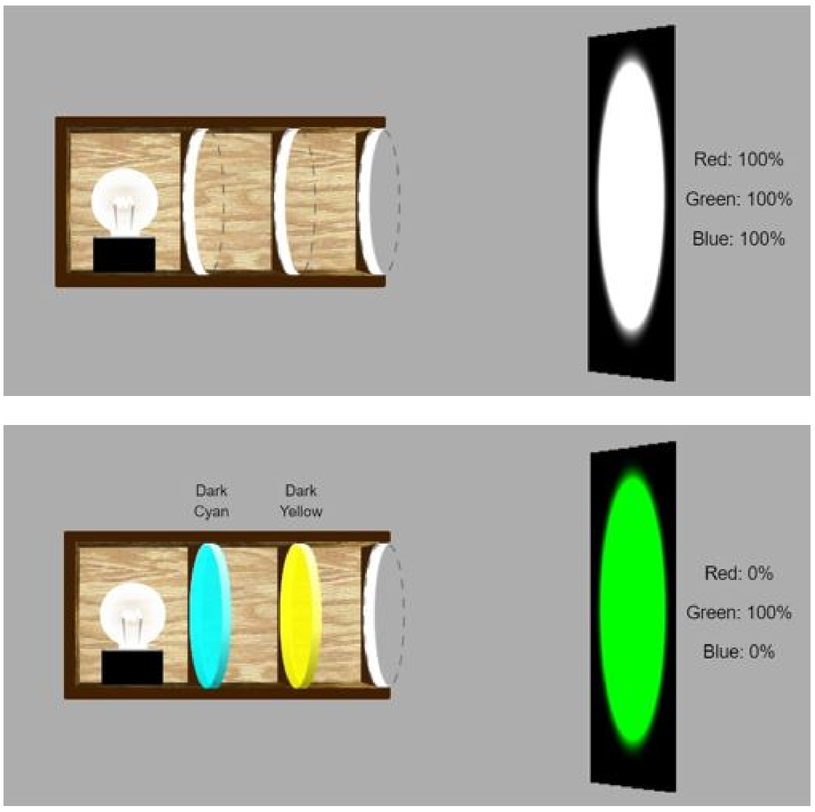


Figure 2.6. *CMY color filtering.*

Therefore, the CMY model is used as a filter for an image. As described before, if the robot wants to separate green objects from an image, putting on a cyan layer and yellow layer filters out red and blue lights so that only green light can pass through those layers - Figure 2.6.

- **HSV model** The two models presented above are oriented for computer or robot which means they are presenting color in hard way. In contrast, the HSV model, which includes in its name Hue, Saturation and Value parameters, displays more details based on the perception of color.

  HSV model is usually demonstrated as system below, where H represents the base color, S is the intensity of base color and V (in other documents, V - Value also known as B - Brightness) returns the shades of base color. By tuning these three parameters, the target color can be achieved.

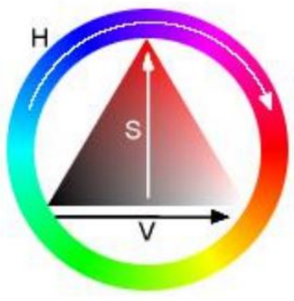


Figure 2.7. *HSV model system.*

In this work, HSV model is used as filter for object detection. Beside color detection, the objects need to be shaped so they can be detected. As the result of color filtering, not only the objects are highlighted but other disturbances with similar color related could be accidentally revealed also. Next part discusses more about contour detection, which is implemented into HSV model to filter out image and capture only the desired object instead of everything.

Image Segmentation

As mentioned before, an image can be represented by a function called *image function*. This function, in general, is defined as a vector function which contains numerical values of one or more technical details belonged to the pixel. In this case, the image is captured by color mode camera so that the *image function* is generated from pixel coordinates (X_I, Y_I) with three parts as $I_r(X_I, Y_I)$, $I_g(X_I, Y_I)$, $I_b(X_I, Y_I)$ with respected to three primitive colors of RGB color channel. On other hand, if the image is captured by mono-color camera, the *image function* is generated with only one part from pixel coordinates as $I(X_I, Y_I)$. This parameter from black-and-white image is the representation of the grayscale diagonal in RGB cube, in other words, it is the light intensity display of pixels.

Image segmentation is one step in image processing, where picture is divided into a certain number of parts or regions, which is called *segments*, that each member pixel of a region has similarity in one or more properties compared to other members. Typically, each region of an image is the extraction of objects from the surroundings or homogeneous regions in the environment.

The problem of image segmentation can be solve by two popular approach and they are region-based segmentation and boundary-based segmentation. Generally, the problems of image segmentation is inconsequential when there exist many proposals beside the twos mentioned above. However, this thesis work mainly uses the results from those two methods, which are discussed in detail later. Besides, these two approaches are not completely different from each other. After using region-based segmentation, the contours of result can be detached and used as result of boundary-based method and in vice versa, the pixels, which are inside the closed boundary, can be archived as result of region-based method.

- **Region-based segmentation:** The concept of this method is grouping collectively those pixels, which share common characteristics. To do so, the algorithm starts with random initial groups of pixels the growing those groups by connecting other neighbor pixels if it identifies that the neighbors have same values in considered properties as the groups. In many operations of rational works, the region-based approach is used to segment object from others by the *grayscale* value. A good threshold is the key for satisfying result with this approach. Because the *grayscale* can be presented as 0 and 1, this method is also called *binary segmentation*. In additions, the result of this method can be improved by using multi-spectral images.
- **Boundary-based segmentation:** The key idea underlying boundary-based segmentation techniques is that obtaining number of many single local edges, matching with the gap of light intensity of image. The local edges, in other words, are the collection of pixels so the boundary-based segmentation can be simple explain as detecting continuous pixels at light intensity gap of an image.

As can be seen, the memory usage of Boundary-based segmentation is smaller than Region-based segmentation due to a reduction of pixels storage. However, from computational perspective, the region-based method is probably better because the task of similarity checking is more simple than detecting the light intensity differences and connecting those pixels. Besides, the results of boundary-based segmentation is not always a closed boundary while it can be obtained from result of region-based method.

2.1.2 Visual Servoing

Occasionally, the robot operates in self-adjustment under human supervision. In this thesis, the self-adjustment system works based on visual measurements. These data provide the robot the ability to model the surrounding environment for its task to control the end-effector to be at correct position in respect of the object location observed by a camera in real time and do the pick/place action.

The problem of Visual Servoing approach is that the measurements from camera are directly transferred to robot processor. Those data represent for features of an 2D image plane while the robot works in a real operational space and the purpose of Visual Servoing is reaching and keeping an target pose with respect to observed object. There exist two kinds of control solutions which are *position-based visual servoing* and *image-based visual servoing* [41]. Each approach has its pros and cons. In either case, the position of camera or camera calibration is an important issue that causes the difference between two methods. Obviously, the *position-based visual servoing* requires more sensitivity in calibration compared to the other method. On the other hand, *image-based visual servoing* method produces data directly in image plane and camera calibration in this situation could be considered as device disturbances and reduced later.

Besides, when comparing pose estimation function, the second method gets more credits than the eye-to-hand camera position approach. The *position-based* approach requires object geometry for pose estimation task if it is used with mono-camera system because the information about the object is gained from many perspective of the camera while the robot is moving. In contrast, *image-based* method does not require object geometry to do pose estimation task even the method is working with single camera.

Under the thesis circumstance, *image-based visual servoing* is utilized due to the simplicity in setting experiment work space. The control approach with *image-based visual servoing* is calculated on the basis of differences between current states of the robot and the feedback from camera. It is worth noticing that the data from both sides are needed to compute in the same perspective. Usually, the robot arm already has transformation between each joint and link to the base of the arm. To simplify the perspective transformation task, the coordinates of camera position in work space can be described as a static joint of the robot that link with the base. In that case, the pose of object estimated through camera can be performed as the location of robot arm base coordinates. The control scheme of this approach is described in Figure 2.8.

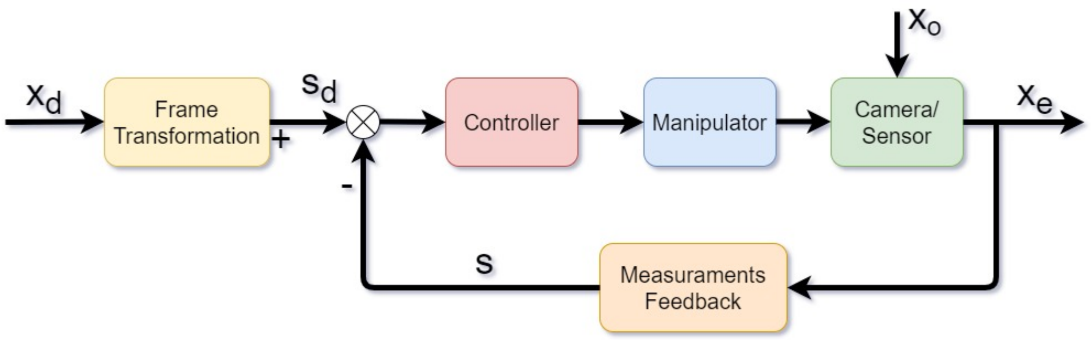


Figure 2.8. *General block diagram of vision-based visual servoing.*

2.2 Ontology

After getting raw data from previous section, one needs to organize these data into a meaningful representation. The ontology is an effective way to do that. Knowledge base plays the main role that enables the robotic manipulators to productively accomplish the service tasks in real environments. However, the knowledge base, which contains structured information, is developed in different ways due to the variety of practical problems. In this thesis, the information is collected from many sources which are in different platform. The visual data gained from Kinect camera is one example. Similarly, UR5 manipulator internal data and interaction data collected from web user interface are instances of such multi-platform information also. Hence, the question of this section is how to represent data when the robot is under processing and how to manage the data as a knowledge source, based on which the robot learns and generate actions accordingly. In additions, this thesis model is a simple task with manipulator, which is related to some previous existing robotics knowledge bases such as Willow Garage Household Objects Database [10] and the Semantic Object Maps (SOMs). The database from Willow Garage is designed as a lite SQL database and the SOMs database, which is introduced by Rusu et al in [37], is more complex than the Household, where it not only stores information about the characteristics of the predefined objects including their 3D CAD, but also stores their spatial data for grasping purposes. From the above reasons, the knowledge base should be illustrated as a semantic graph or network which have ability to link information between concepts. As a result, a suitable structure is an ontology [28].

In computer science, ontology is a structure of data to describe fields and mostly used to make deduction about objects in the mentioned fields and relationship among them. Ontology provides a common vocabulary set of concepts, important attributes and the definitions of former concepts and attributes. In prior to common vocabulary set, the ontology also provides constrains, which are sometimes known as fundamental meaning of the vocabulary set, to use in communication among human and other complex distributed application. Particularly, this thesis uses the ontology as the common knowledge base for communication between different platforms as ROS, Linux computer and Web application.

For modelling the world or just a simple view of the world, the ontology is the best option to store the scenarios that are linked to each other for the semantic query. Correspondingly, it is used as a standard framework of knowledge presenting which is architectured as:

- **Individuals**: the subjects that ontology describes.
- **Classes**: the labels of concepts, collections that draws the common features of individuals.
- **Properties**: features, characteristics or parameters.
- **Relation**: links between classes or individuals.

The ontology vocabulary sets mentioned above are built from infrastructure of Resource

Description Framework (RDF) with rules and relations defined in Resource Description Framework Schema (RDFS), which are the universal languages for introducing information of web resources and supporting semantic reasoning.

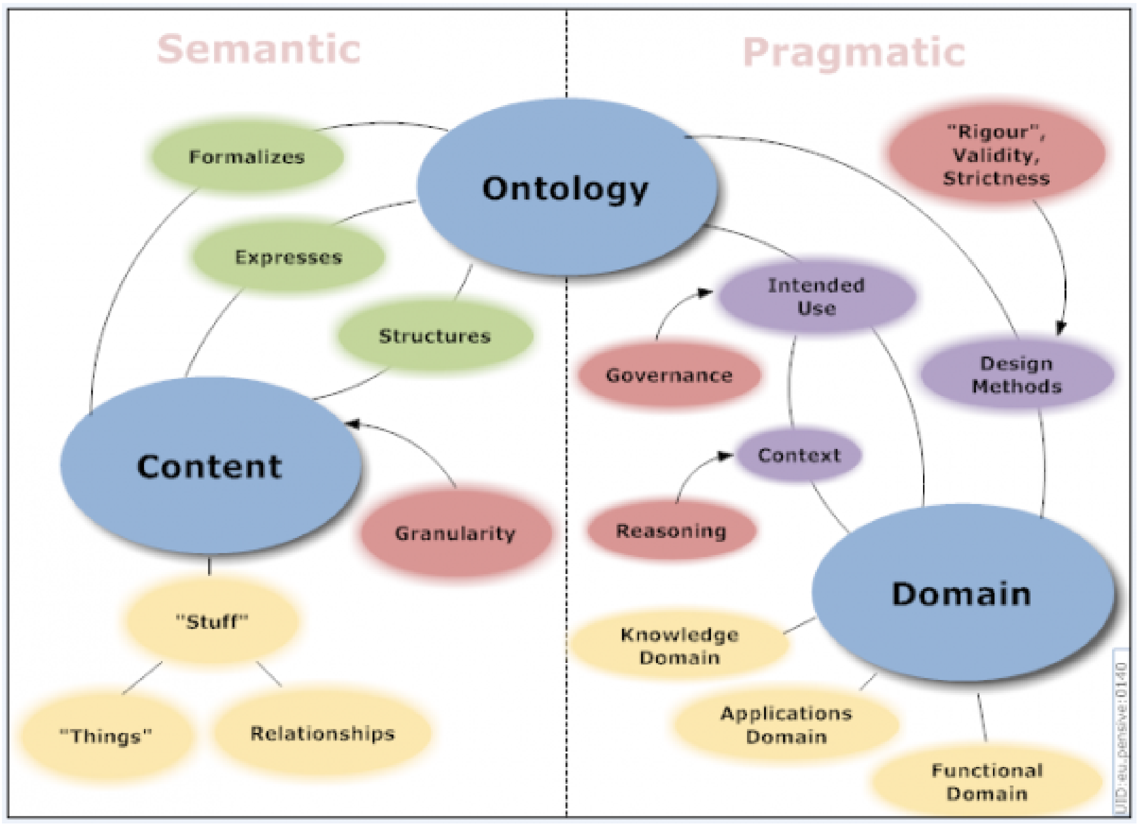


Figure 2.9. *Ontology definition map [47]*

2.2.1 Ontology Components

Individuals

An individual is one of the fundamental elements of an ontology. Individuals of an ontology could include specific object such as human, animal, tool, table, mechanic parts, etc... and also include abstract object as words or other individuals. The ontology has no dependence on the individuals but in most circumstances the purpose of an ontology is providing meaning of individual classifications despite the fact that those individuals are not parts of mentioned ontology.

Classes

Classes can be defined as collections, sets of abstract objects such as common attributes, parameters to describe the feature of each class. Besides, the class may include instances or other classes, which means it has the ability to include the combination between instance and class as well because a class can play role as an instance. The ontology is divided into individuals and base on the characteristics of individual, they are labeled to suitable class.

Properties

The objects in an ontology could be described by declaring properties of them. Each property always has its name and default value. The properties in ontology are the places where real object characteristics are stored. To clarify, a person could have information of Name, Date of Birth and ID hence in an ontology, an individual, which represents for that person, also has properties as "name", "date_of_birth", "ID" to store those information. The value of property has a wide range of datatype, in other words, to describe a mechanic part in the digital platform, the property can stores the 3D CAD model of that part.

Relation

One of the main applications of using properties is describing relations among individuals in ontology. A relation is a property that its value contains an object of ontology. In various types, an important relation is subsumption. This relation illustrates the link between individuals and classes. Nowadays, the communication between platforms using ontology as a shared knowledge base is an ambitious goal that requires the vocabulary sets of ontology, which should be as common and easy to comprehend in most circumstances. For further explanation, a robot is developed to able to process command such as "grasp the cup", "assemble these parts" and perform those tasks, which are stored in ontology and linked to code scripts that control the robot to do the requirement.

2.2.2 Ontology layers

Developing ontology is a time-consuming task despite the fact that declaring classes, properties, individuals is uncomplicated to accomplish. Those definitions need strong connections among them to make ontology vocabulary work properly. As a consequence, when the ontology is the source for multi-users, all misunderstanding should be avoided. That is also the reason why the ontology is usually used to demonstrate a specific field of interest. Although every ontology describes many different topics but they are in the same structure which includes a generic layer, a middle one and a most specific one.

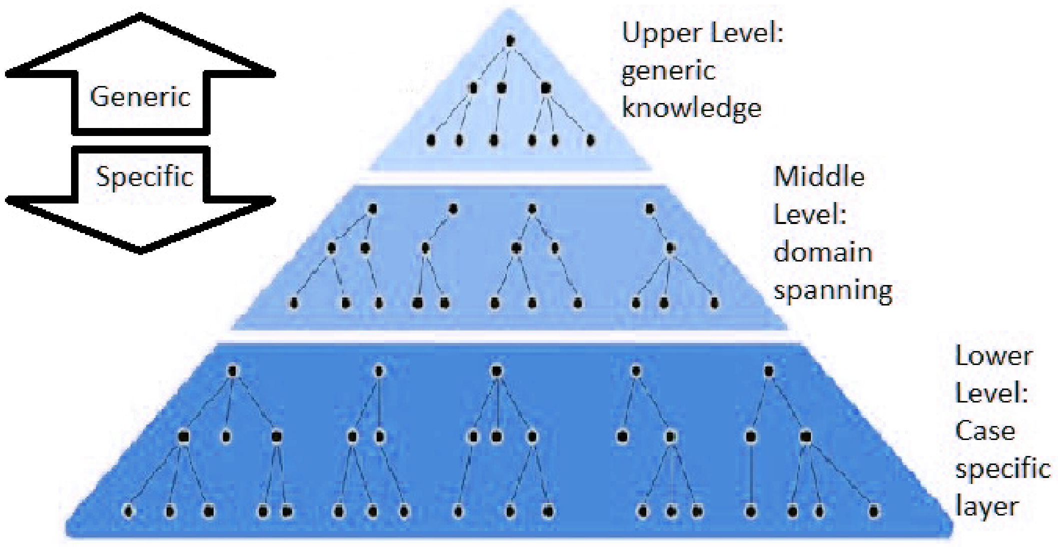

Generic
Specific
Upper Level: generic knowledge
Middle Level: domain spanning
Lower Level: Case specific layer

Figure 2.10. *Ontology pyramid*

Upper Ontology

Upper ontology is the most generic layer of ontology. In most circumstances, the top level of ontology contains highly abstract concepts or objects which could be time, events or identities. This makes the upper level a very important part in an ontology. The ontology is developed from top to bottom of above structure pyramid; therefore, the upper level could be difficult to build at first. Next, the concepts of upper ontology would be expanded with more clarified information in middle level.

Middle Ontology

In the middle level, concepts and objects, as explained in the previous part, to some extend, are related to upper ontology. They are also reusable as their definitions are not too specific about the case of interest that ontology is describing. However, it is hard to draw a separate line between the upper and middle level because their definitions depend on how the ontology designer describe what is generic knowledge and what is case specific knowledge. Last but not least, linking both mentioned levels to the case of interest is the a big challenge for an ontology designer and that is where the final layer happened.

Lower Ontology

This level contains specific concepts and objects used to describe the case the ontology covers. In this level, the data in interesting circumstance are actually stored and processed. Provided that, this is the biggest part of an ontology. Usually when the ontology is on operation, the lower level is called and processed with high probability that the data grow bigger and bigger over time, which poses a risk to the work flow if there is some inconsistency between lower level's concepts. For that, developing lower ontology is a time-consuming task.

Here is an example for ontology layers. In this case, the designer is developing an ontology that describes a Festo automation device.

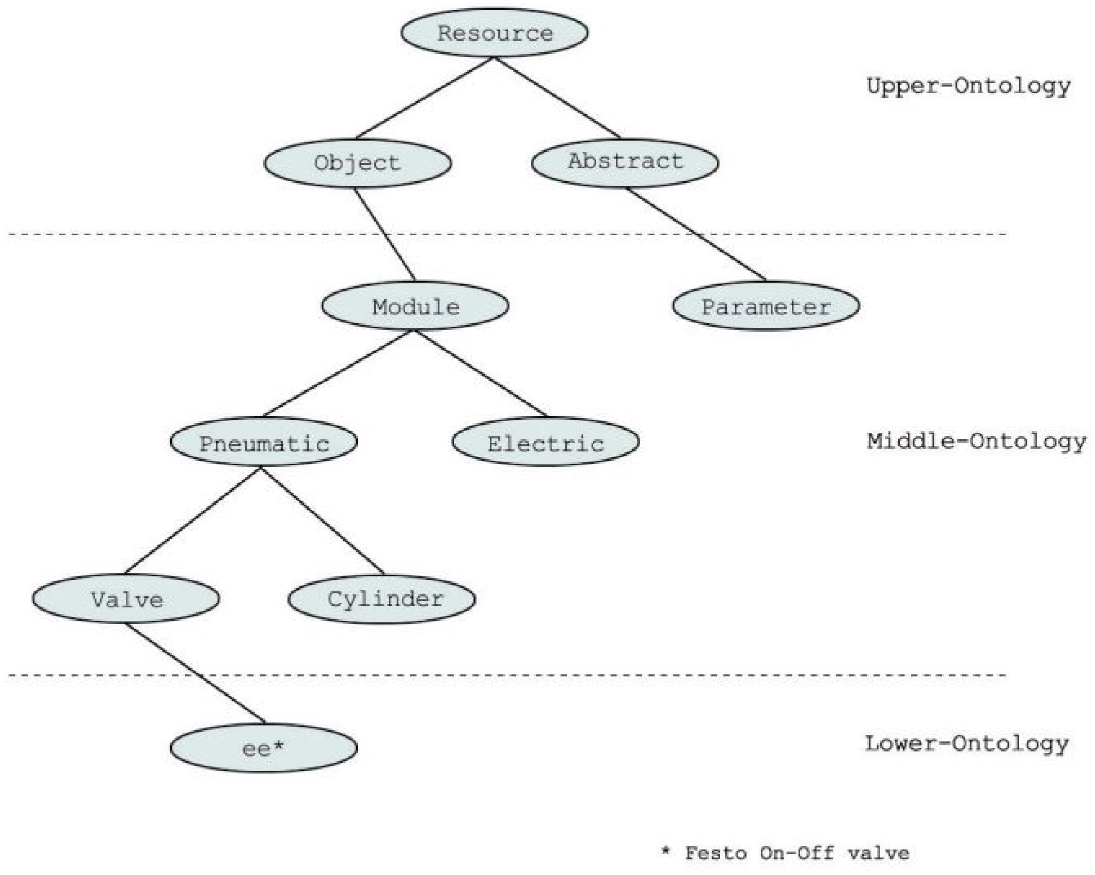


Figure 2.11. *Ontology layer example [31]*

2.2.3 Ontology Language

In general, the ontology is only the concept or definition about object or individual so implementing it to a system is a task that requires teaching or communication to make the system understand the concept. Language is a traditional way to do that. In this circumstance, language is programming language which is used by human to create communication pipeline with machine. The history of ontology language has begun since 1980s and archive interesting of the world in late 1990s until today. There are various programming languages could be used to present ontology but in the boundary of this thesis, OWL is the best option due to the ontology using in this scope was developed in Protégé editor from Stanford University.

More details of developing ontology and how to integrate ontology in this thesis is presented in next chapter. Below, brief information of languages that help constructing a complete ontology are presented.

Web Ontology Language - OWL

The ontology vocabulary, as previously, could be developed from RDF and RDFS. Albeit, they just provide the tool to create definitions and define the relations between classes without the ability to add information. To solve that problem, OWL, which is the abbreviation for Web Ontology Language, meets that requirement. With history of endorsing and publishing in early 2000s, OWL is now popular and well instructed for users. Evenly, new users could be comfortable with instructions of OWL since W3C had published six separated documents to explain and user guide in February 2004.

OWL versions

Presently, there are three versions of OWL in use worldwide. Those versions are listed below.

1. **OWL Lite** supports simple tasks and requires only classification hierarchy and simple relation between classes. OWL Lite has a limitation that it only supports cardinality constraints with two values of 1 and 0. As the name of it, this version is the simplest version of OWL.
2. **OWL DL (Description Logic)** has all OWL language architects, thus supports more complex tasks with full options of presentation but still has the ability of computation of conclusions. Albeit, the components of OWL DL could only be used in restricted conditions as a class cannot be a property or an individual and vice versa. This version got its name based on its original project, which is a logic field of research that contributes to the formal OWL foundation.
3. **OWL Full** It is a full version of OWL with maximum expressiveness but no guarantee for computational ability of conclusions. With maximum expressiveness, the OWL Full has ability to simultaneously play role of an ontology, which is collection of objects and concepts, and role of an object to itself. Hence, this complexity leads to a reality that it is unlikely for most reasoning softwares to be able to support OWL Full version.

OWL basics

In other words, OWL is the extension of RDF so it inherits the basics of RDF also. The cell elements of OWL are classes, properties and individuals or instances. These elements are defined as built-in ontology vocabulary under namespace *owl*. Because the full version of OWL has maximum expressiveness and it is difficult to discuss in detail, the next part focuses mainly on the basics of OWL Lite and OWL DL.

- **Classes**
- **Individuals**
- **Properties**
- **Property Restrictions**
- **Property Characteristics**
- **Boolean Combination**
- **Enumerations**

2.2.4 Reasons to use Ontology

From the beginning, the Ontology concept is related to a philosophy where it concludes the properties of problems or objects which is used as common knowledge in multi-field researches. However, the rocketing development of technology with significant example of Artificial Intelligent (AI) gives popularity to Ontology concept. Nowadays, Ontology can be discussed between web developer or robotic engineer as a tool as common language.

There are several reasons to use and build an ontology as mentions of Natalya and Deborah in their previous work [32].

- *First* The "understanding" is main purpose of an information structure and with ontology, "understanding" is the goal between human and machine intelligence. In other words, instead of saying robot-like command as "Go to coordinates X 20 Y 10 Z 2" to an assistance robot, we could give a human-like command as "Go to the Kitchen". The robot can look up "the Kitchen" in their knowledge base and check the coordinates it should move to. Another example, there are many database about medical information or health services. If these database could be shared or published based on same ontology vocabulary, so not only medical applications but also human could query these data to increase effectiveness in medical and health care activities.
- *Second* Ontology provides the ability to re-use knowledge domains.
- *Third* Developing ontology also creates clear hypothesised definition domains.
- *Fourth* Ontology allows separate knowledge domain from execution domain. As example of above circumstance, the knowledge domain of robot defines what "the

Kitchen" mean to it and execution domain process that "understanding" as machine knowledge which is related to technical parameters like coordinates.

- *Fifth* Domain knowledge analysis is possible when the definitions of concepts in the ontology are known.

2.3 Discussion

In summary, a brief introduction of ontology, its benefits and components has been given in this chapter to justify the author's choice to use ontology in this thesis work. Its structure and pathway to develop an ontology is also explained. Robot vision field is also explored as the main tool to successfully perform the planned task. After discussing what robot vision is, the second part of this chapter shortly presents two important components of robot vision, which are image processing and visual servoing for handling the input image from camera and using the image features to control the robot accordingly. From the discussion in this chapter, the key underlying idea of this thesis work is extracting features from captured image then storing the features as structured data in ontology as a knowledge base that a reasoner can query to comprehend the environment and control the robot by those visual measurements.

3 RESEARCH METHODOLOGY AND MATERIAL

The main goal of this thesis is to make a robotic manipulator perceive and manipulate the surrounding task space by combining multi-platform blocks that share a common knowledge base. As the system implementation is the result of a multi-platform configuration, this chapter is divided in to four sections representing for the four main components, i.e, ontology, object detection, ROS environment with the web user interface and communication between multi-platforms.

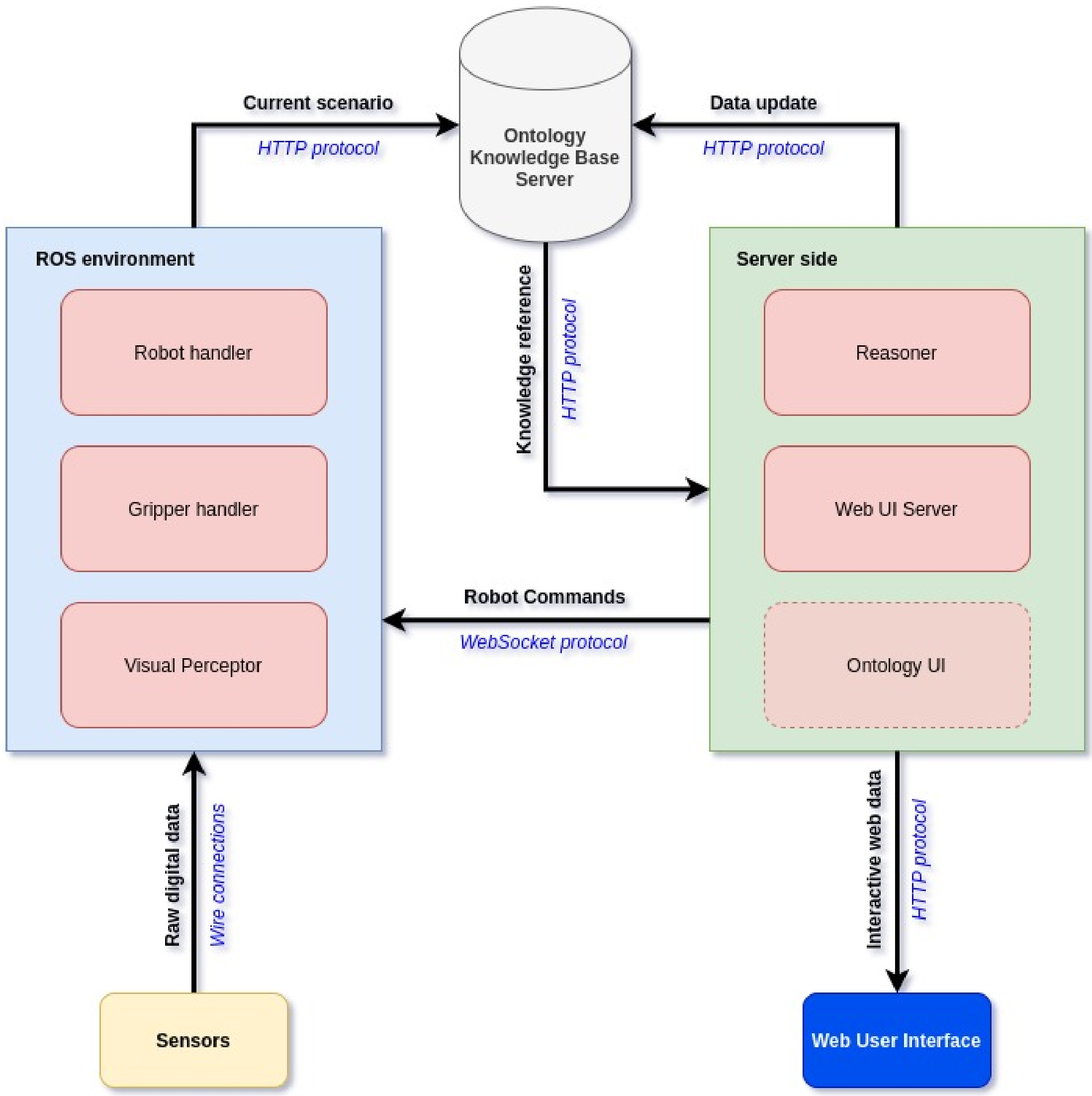


Figure 3.1. *General diagram represents the thesis system.*

The structure of this chapter is similar to structure of actual system, which is aiming to represent a framework for a knowledge-based manipulation. The diagram of Figure 3.1 describes the system architecture in an intuitive way.

The system process starts with the raw data from the sensors. The sensors used in this work are stereo cameras (the Microsoft Kinect v1 and Intel Realsense d450 were used in two experiments), internal joints sensors of UR5 manipulator and an on-load sensor of vacuum gripper. The collected data from sensors will be handled by three independent units running in ROS environment before updating the ontology server. The server side block consists of three different units, where reasoner is the most crucial unit among others. This prime component look into current conditions of surrounding environment including the robot and gripper status then generate the reasonable actions of the robot to accomplish the task. Furthermore, the reasoner also receives the command from users via user interface for both updating knowledge base and accepting mission. The desired outcome is to obtain the spatial data of objects and complete a desired scene by grasping and releasing the detected objects. Another unit in this block is the web UI sever. This unit controls the web, what is used as Web User Interface in the diagram, by getting requests from both operator and the reasoner then visualizing responses on website. Nowadays, website is chosen for being the user interface as it is a friendly, and easy-to-access tool for human to interact with. In additions, the dash-line unit, i.e, Ontology UI is optional to use to modify the ontology server. Last but not least, the arrows represent the connection between different blocks. The black arrow represents the transmission data while the blue one specifies the type of connection that is used to transmit data.

The first section is dedicated to explain the definition of ontology and how the knowledge base in this thesis is developed with Protégé software. Second section introduces the detection task that has duty to segment the real time input image from camera then extract the needed information for storage. Next, the environment or so called the *platform* is first time described in this document. ROS is well-known as the biggest open-source community of robot developers while Gazebo is a good physics simulation world that is compatible with ROS and has strong society also. Like other automatic machine system, the setting parts are well assembled and a human-machine interaction instrument is required for safety reasons, in case, the system needs an interruption when it get errors. Furthermore, the interaction with human, in this situation, could help the system produce more efficient and become friendly while the knowledge of robot could be display in the way that human understands. Last but not least, the communication pipeline, which is the data transmission between groups in Figure 3.1, must be brought in the conversation.

3.1 Ontology Developing

As mentioned in Chapter 2, the task of developing an ontology is an excursion of ontology life-cycle and at every stop, the information in the field of interest is expanded with more details and constraints. First of all, the domain and scope of the ontology should be considered. Because the case of interest is the current situation that the robot is working under, the domain has role to represent the environment status. This thesis aims to use ontology to detects and adjusts the differences between the target scene and the current scene. There are lots of methods to define an ontology and one of those is listing a series of questions which represent the systems. Via these questions, the ontology should have the ability to understand the representation of the surrounding world, which in turns reason the desired solution. This approach is called *Competency questions* in [32].

This work adopts the mentioned approach to represent the system with the following Competency questions. **Competency questions list**

- What is the robots and where did we put them?
- What is the grippers and where are they?
- What is the sensors and where did we place them?
- What is the object and where are they?
- How to control robot R1 of class robots?
- How to control gripper G1 of class grippers?
- What do the robot R1 do to finish the target scene?
- Is the gripper R1 ready to pick object O1?
- Is the gripper R1 ready to place object O1?
- Is the camera C1 ready to observe?
- What is the difference between current scene and target scene?

From these questions, the ontology could understand the system and provide necessary information such as the status of robots, grippers and give an alert if the gripper is not ready for the picking task. The detail structure of in-use ontology are determined as follow (this structure is built based on Protégé Ontology Developing documents).

Classes and Individuals

- Robot: UR5
- Gripper: Vacuum_gripper
- Sensor: Kinect
- Object: individual of this class is updated at run time
- Task: Pick, Place, Scan, Move, Completing_Scene

Properties

1. **Object properties**:
   - has : this property shows that the object has ability to do something.
   - requires : requirements of the objects.

2. **Data properties**:
    - Initial_state.
    - Current_state.
    - Data.
    - Position.
    - Status.

One last prime components of an ontology is the **Relations**. It is convoluted to present the links between classes and others by just text so the constrains of knowledge base is marked as Figure 3.2.

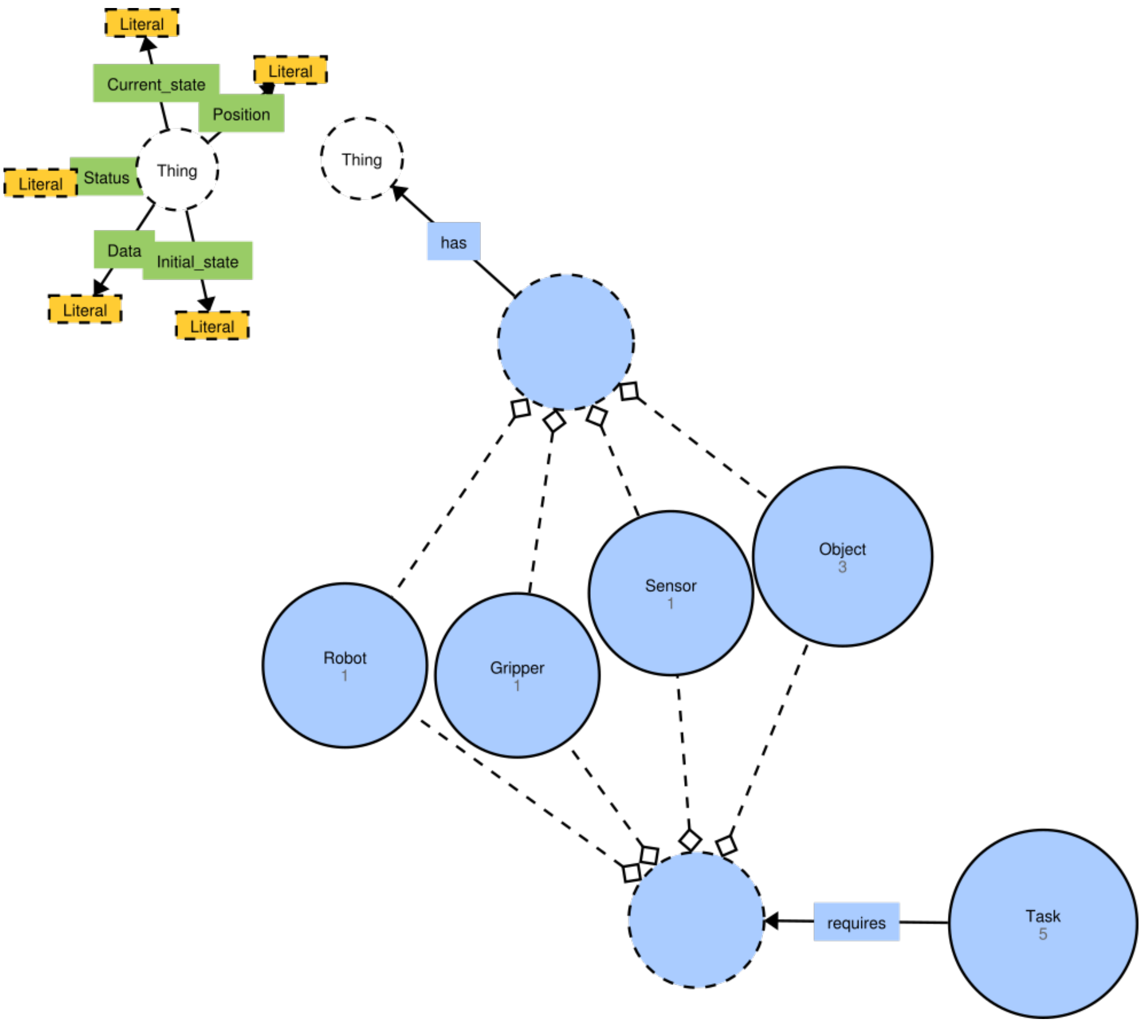


Figure 3.2. *Knowledge Base visualization.*

From the image above, The figure is annotated as Blue Circle for Class, White Circle for Individual, Blue Retangle for Object property, Green Rectangle for Data property and Yellow Rectangle: Value of the mentioned Property. The definition of done in ontology task is that the knowledge base has ability to provide answers for the query of **Competency Questions** list. Correspondingly, the graph is the visualization of solutions for questions list above so the ontology developing task gets accomplished. For more details, the untitled dashed circle class is the union of classes, which means if something is related to a union class then that thing is linked to one or more member classes of the union. It is obvious to say an example from the visualization as "**Task** T1 ***requires*** **Gripper** G1 that ***has*** **Property (Thing)** ***Current_state*** *Open*" with *Open* is the information that gripper sensor collected.

3.1.1 The Reasoner

Although Apache Jena Fuseki server has its own server, a separate reasoner is developed to handle the knowledge base for supporting this thesis purpos and it is designed to be an independent module of the proposed system. For short, "The reasoner" is termed for the separate reasoner because the thesis work does not use the built-in reasoner from Apache. The relation between the reasoner and the knowledge base is query-response process. The reasoner, when it is triggered, queries the knowledge base about the current scene information and the target scene details. As described in Figure 3.2, each individual of the knowledge base has five properties those are "Current_state", "Status", "Position", "Data" and "Initial_state". For example, the "Position" property of "Kinect" instance is the coordinates of Kinect camera in experiment setting and these coordinates are constant while the same property of "Vacuum_gripper" is the current coordinates of the gripper and the coordinates of gripper are updated when the robot is moving. Because the individuals in ontology are structured for easily query purpose, there are specific queries defined for querying task. Defining specific queries allows the queries to be reused and it helps to simplify the code of the reasoner.

The reasoner has two main duties: detecting the differences between the current scene that robot is seeing with the scene that user defined in knowledge base and extracting the suitable actions to manipulate objects to complete the target scene from current scene. To accomplish those tasks, the number of object types, amount of them in each type and their estimated position on table are considered. The individual of class "Object" is created at scanning processing when there is new detected object. These individuals are representations of current scene objects. The object information of target scene is contained in "Data" property of "Completing_Scene" task. With this structure, the "Initial_state" property of "Completing_Scene" task stores only name list of target scene's objects while the "Current_state" contains the name list of current objects on table. The reasoner goes through the list of objects in both scenes and start the processing as Figure 3.3.

There are actions blocks presented with colors in Figure 3.3. The green block represents for accepted objects which means there is no action needed. The yellow block shows that the robot should modify these objects to get correct position. The red block is for removing task if there is redundant objects that they need to be removed from the current scene. The last action block is the blue one of adding missing objects task. After the

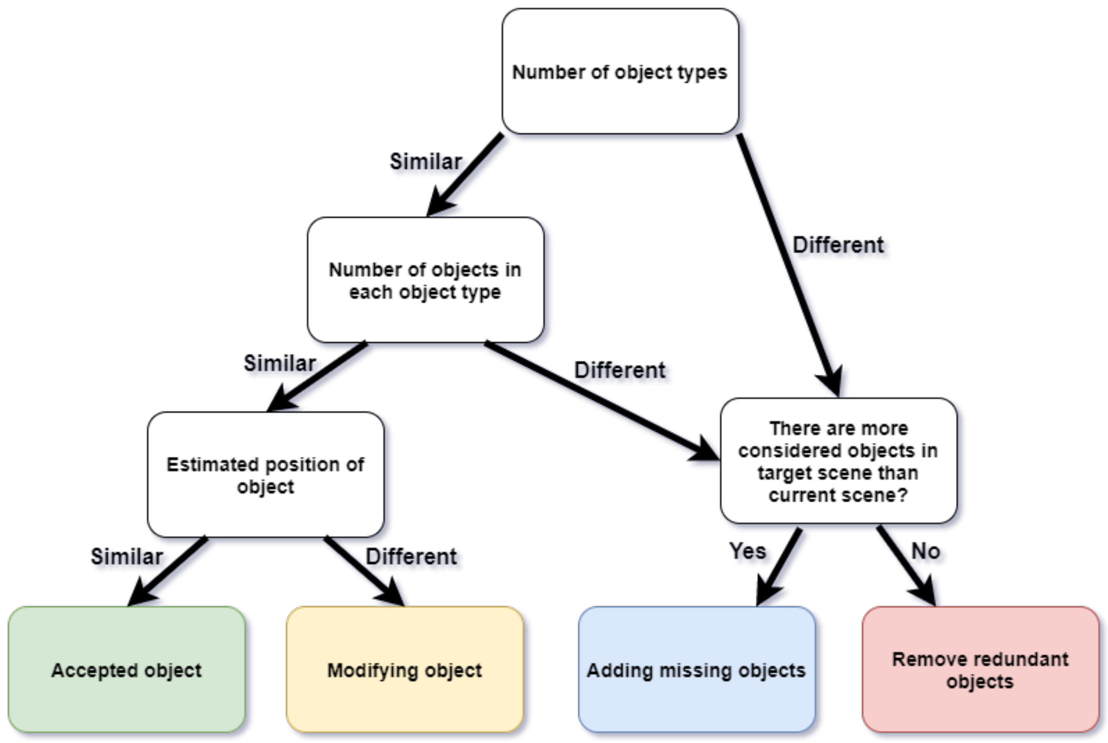


Figure 3.3. *Processing of the Reasoner.*

suitable action attached with each object is performed, the individuals of "Object" class are updated with current situation.

3.1.2 Protégé

From a developing project of Stanford Center for Biomedical Informatics Research, Protégé becomes a productive open-source software, which provides uncomplicated utilizing tools to construct knowledge-based applications and domain models with ontologies.

One of the advantages of Protege is that it allows users to easily make modifications if the system changes. Significantly, Protégé is a good environment to do that where it is easy for developer to interact with ontology and it also includes a reasoner for query testing. After having an knowledge base, the consequence step is is to have the knowledge base online as a server so that other clients of the system can connect and exchange information.

3.1.3 Apache Jena Fuseki

Ontology server usually works as a standalone station. The knowledge base server used in this thesis is designed to work independently because the main purpose of the thesis is to modulize the system (shown in Figure 3.1. The task of developing a ontology server

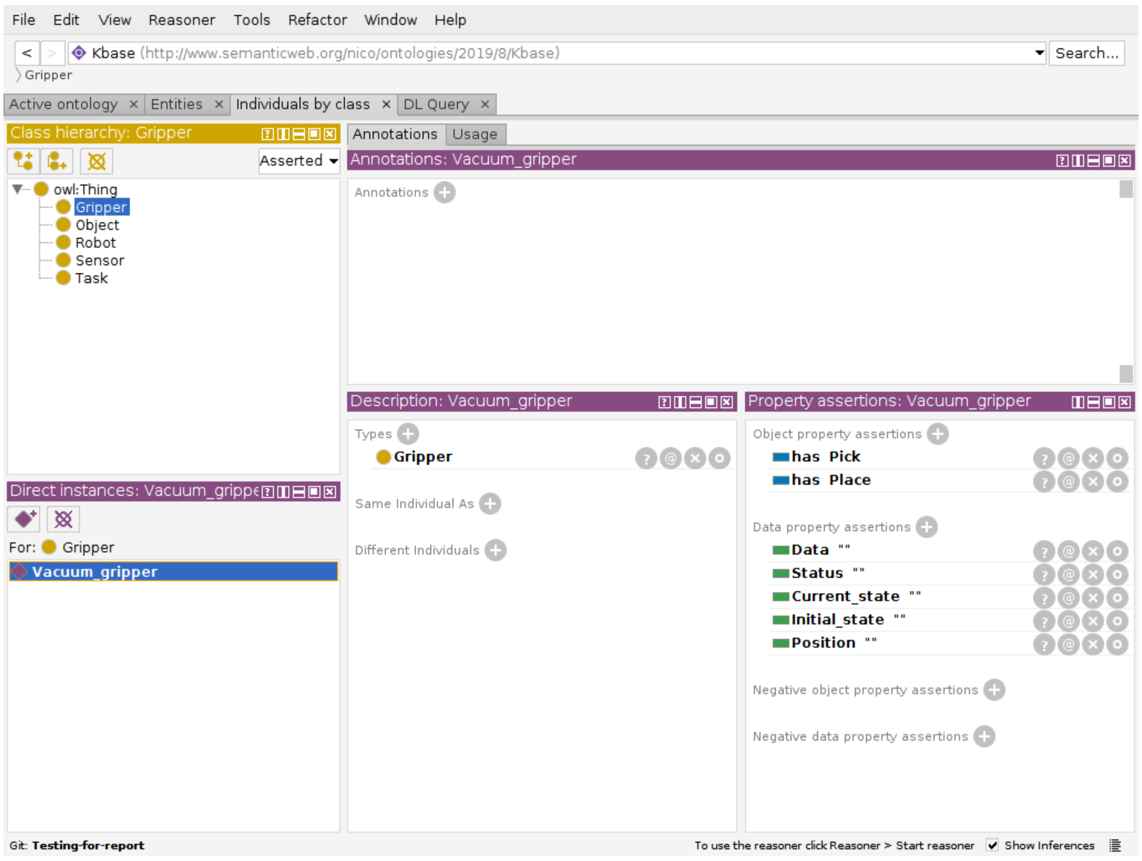

File Edit View Reasoner Tools Refactor Window Help
Kbase (http://www.semanticweb.org/nico/ontologies/2019/8/Kbase)
Search...
Gripper
Active ontology
Entities
Individuals by class
DL Query
Class hierarchy: Gripper
Asserted
owl:Thing
Gripper
Object
Robot
Sensor
Task
Annotations
Usage
Annotations: Vacuum_gripper
Annotations
Description: Vacuum_gripper
Types
Gripper
Same Individual As
Different Individuals
Property assertions: Vacuum_gripper
Object property assertions
has Pick
has Place
Data property assertions
Data ""
Status ""
Current_state ""
Initial_state ""
Position ""
Negative object property assertions
Negative data property assertions
Direct instances: Vacuum_gripper
For: Gripper
Vacuum_gripper
Git: Testing-for-report
To use the reasoner click Reasoner > Start reasoner
Show Inferences

Figure 3.4. *Protégé user interface.*

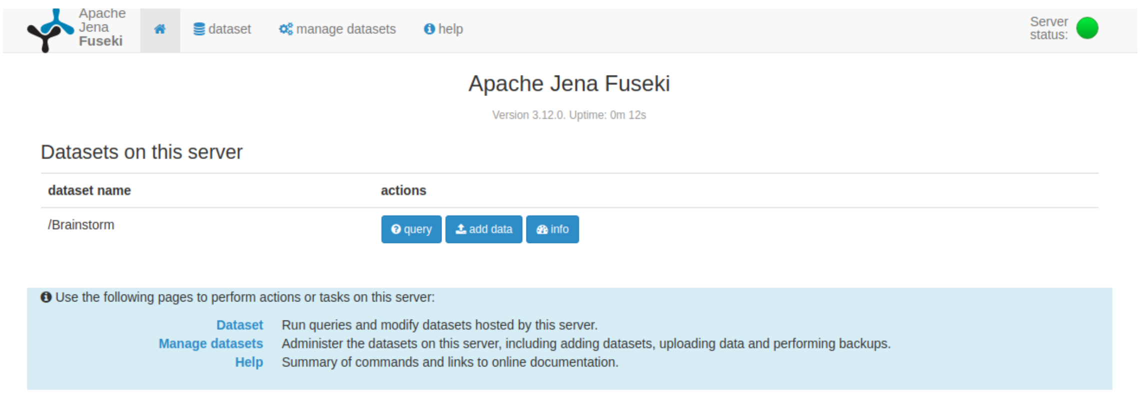
is cumbersome and out of the scope of the thesis. Thus, an off-the-shelf ontology server, i.e., Apache Jena Fuseki is used in this work. This Fuseki software is actually a SPARQL server that belongs to Jena project and it is developed by Apache company.

SPARQL server

SPARQL is a semantic query language that is able to handle data stored in RDF format. SPARQL over HTTP set of command line scripts are embedded in Apache Jena Fuseki and it allows Fuseki server to work with SPARQL 1.1 for processing RDF query through HTTP protocol. The convenience of using SPARQL server is the simplify of syntax. With the simplify syntax, the queries can be defined and reused in many case by changing the considered variable. In this thesis, the queries of knowledge base is defined inside the reasoner code. When the reasoner need to query information in the knowledge base, it automatic chooses the suitable form and fills the desire variable in that form before sending query to SPARQL server. More details about SPARQL are introduced officially by W3C organization [49]. Figure 3.7 displays the example of RDF query that used for SPARQL server.

The first pros of Apache Jena Fuseki is the user-friendly design. The second benefit is Fuseki support SPARQL query and HTTP protocol which means that the query can be sent from different computers or different robots when they are connected to same network. With Apache Jena Fuseki, the task of operating ontology server only include the following three steps.

Step 1: Run the server The ontology should be run before other processes. Apache Jena Fuseki is equipped with various operation options. Due to the role of knowledge base, the server is called by command line as standalone server.



Apache Jena Fuseki

Version 3.12.0. Uptime: 0m 12s

Datasets on this server

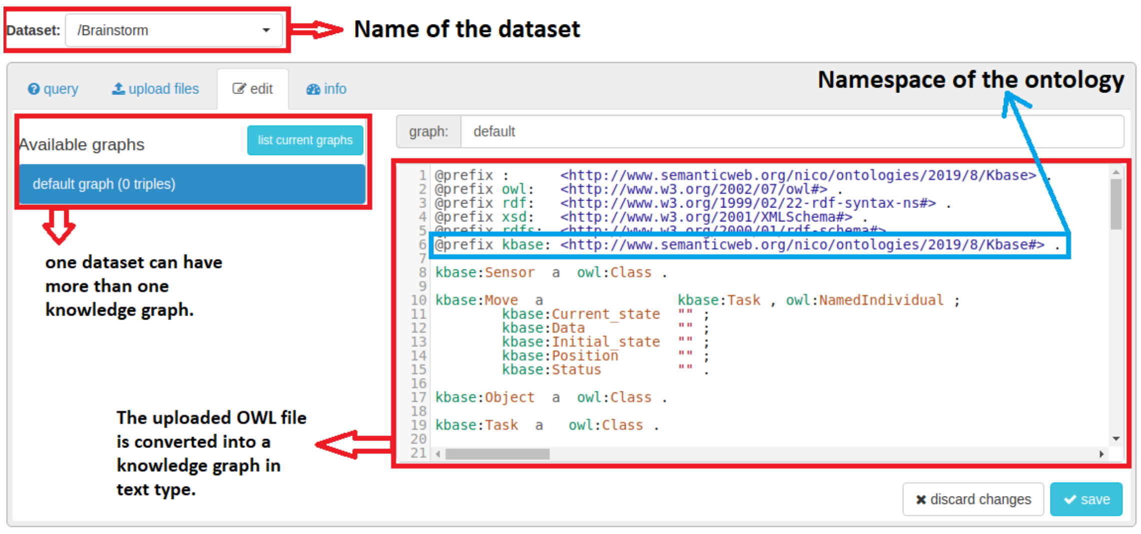
dataset name	actions
/Brainstorm	query / add data / info

Use the following pages to perform actions or tasks on this server:

Dataset	Run queries and modify datasets hosted by this server.
Manage datasets	Administer the datasets on this server, including adding datasets, uploading data and performing backups.
Help	Summary of commands and links to online documentation.

***Figure 3.5.** Fuseki user interface.*

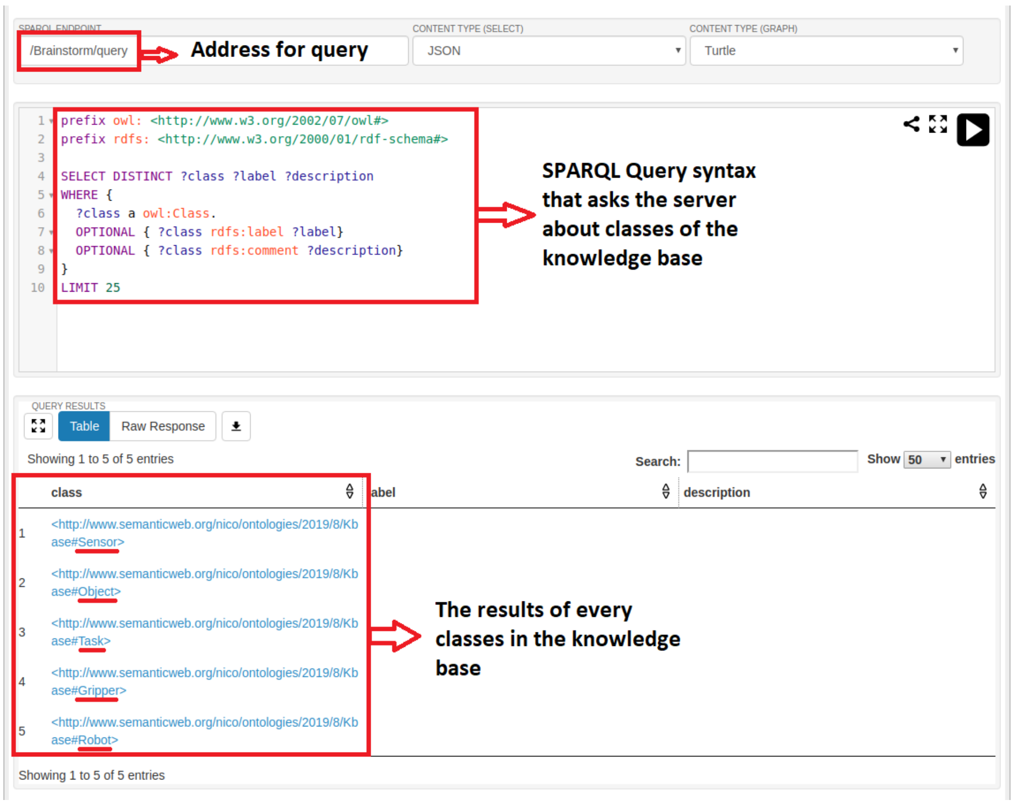
Step 2: Load database There are two ways to upload initial data to the knowledge base. Firstly, there is "Add data" button in home space of Apache Jena Fuseki for user to upload the original ontology to the server. Secondly, if the initial ontology is the same in every tasks so instead of uploading it every time after starting the server, user can download the knowledge graph (the graph is under text type so it can be stored in txt file) and re-use it each time the server restarts. The conversion from ontology to RDF graph is termed as Serialization.


Dataset: /Brainstorm
Name of the dataset
Namespace of the ontology
query
upload files
edit
info
Available graphs
list current graphs
default graph (0 triples)
one dataset can have more than one knowledge graph.
graph: default
The uploaded OWL file is converted into a knowledge graph in text type.
discard changes
save

```
@prefix :      <http://www.semanticweb.org/nico/ontologies/2019/8/Kbase> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix kbase: <http://www.semanticweb.org/nico/ontologies/2019/8/Kbase#> .

kbase:Sensor  a  owl:Class .

kbase:Move  a                kbase:Task , owl:NamedIndividual ;
        kbase:Current_state  "" ;
        kbase:Data           "" ;
        kbase:Initial_state  "" ;
        kbase:Position       "" ;
        kbase:Status         "" .

kbase:Object  a  owl:Class .

kbase:Task  a   owl:Class .
```

***Figure 3.6.** Loading ontology to Fuseki server by using user interface.*

Step 3: Query the database Finally, when the knowledge is stored on the server, user can query the data by sending a query request to *Query Endpoint*, an address that receives and handles the requests of query and has different role with other address used for updating task. The *Query Endpoint* in this work has the name "/Brainstorm/query" where "/Brainstorm" is the name of the knowledge base. For more details, the query syntax and the results from ontology server can be seen in Figure 3.7.


Address for query
SPARQL Query syntax that asks the server about classes of the knowledge base
The results of every classes in the knowledge base

SPARQL ENDPOINT: /Brainstorm/query
CONTENT TYPE (SELECT): JSON
CONTENT TYPE (GRAPH): Turtle

```
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?class ?label ?description
WHERE {
  ?class a owl:Class.
  OPTIONAL { ?class rdfs:label ?label}
  OPTIONAL { ?class rdfs:comment ?description}
}
LIMIT 25
```

QUERY RESULTS

Showing 1 to 5 of 5 entries

	class	label	description
1	<http://www.semanticweb.org/nico/ontologies/2019/8/Kbase#Sensor>		
2	<http://www.semanticweb.org/nico/ontologies/2019/8/Kbase#Object>		
3	<http://www.semanticweb.org/nico/ontologies/2019/8/Kbase#Task>		
4	<http://www.semanticweb.org/nico/ontologies/2019/8/Kbase#Gripper>		
5	<http://www.semanticweb.org/nico/ontologies/2019/8/Kbase#Robot>		

Showing 1 to 5 of 5 entries

Figure 3.7. *Syntax and result when sending a SPARQL query to Fuseki server.*

To make the query can be sent from any devices in the same network to the *Query Endpoint*, we chose to embedded query form in the Reasoner coding script and the value of the query can be replaced for many purposes. This action makes the Reasoner autonomously ask for knowledge and manipulate the action of the robot based on environment changes.

3.2 Object detection

This section describe the "Visual Perceptor" block shown in Figure 3.1. As mentioned, there are two cameras those were used in this work, the Microsoft Kinect and Intel RealSense. They are stereo cameras, which has the ability to provide point cloud data generated from sub-cameras in different perspectives. The object detection in this work is an important part because it has duty to process and supply input data for the rest of the system. However, processing streaming image in real time can slow down the system and cause time conflicts between the Reasoner and the Knowledge Base Server. With the goal of combining technologies towards semantic manipulation, the object detection is improved step-by-step in this work. The small step starts by processing simple cases where the program classifies sample objects from streaming data of camera. This problem will be analysed in next three approaches.

3.2.1 Point Cloud Clustering

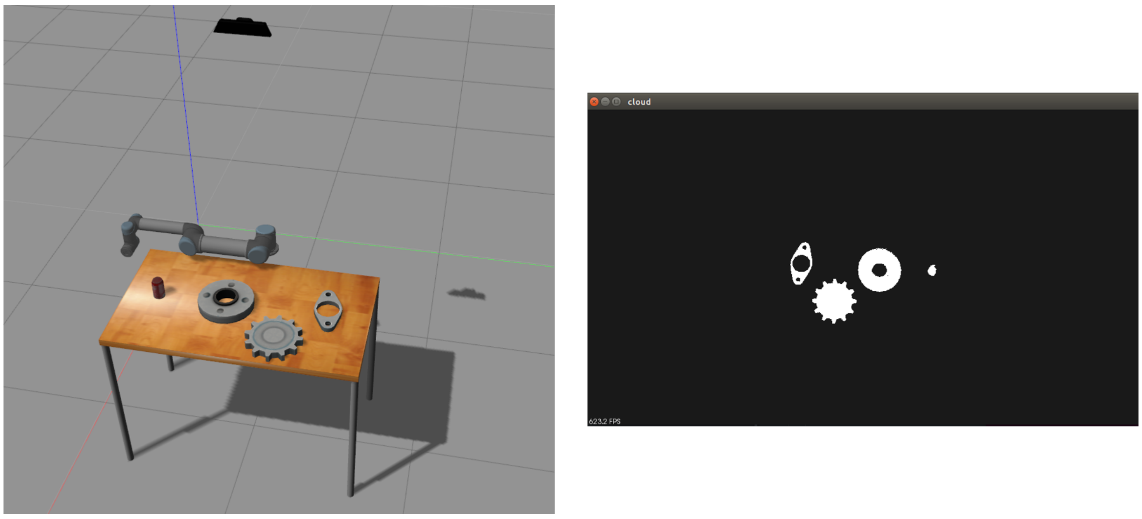


Figure 3.8. *The experiment environment and the result of objects point cloud extracted from the environment.*

Point Cloud Library or PCL is a crucial tool to handle the point cloud data. In this approach, the raw point cloud is collected by the Kinect camera and published in ROS topics. The "Visual Perceptor" block has duty to subscribe to that ROS topic and process the streaming data. In more details, the point cloud data, after coming to "Visual Perceptor" block, is down-sampled and re-structured as a Kd-tree representation for cluster extraction algorithm [36]. The algorithm steps is shown below.

Extraction algorithm steps

```
1. Define Kd-tree structure for input point cloud dataset $P$.
2. Define empty list of clusters $C$ and data queue of processing point $Q$.
3. for every $p_i$ of $P$ do:
    • Add $p_i$ to $Q$.
    • for $p_i$ of $Q$ do:
        – search neighbors set $P_k^i$ of $p_i$ in the sphere with radius $r < d_{th}$.
        – for point $p_k^i$ of $P_k^i$, if it has not been processed, add it to $Q$.
    • If all points in $Q$ are processed, add $Q$ to $C$ and empty $Q$.
4. return point clusters $C$ when all points of $P$ are processed.
```

3.2.2 Image Processing

As mentioned in Chapter 2, the key idea in this approach is detecting main color of objects in HSV color channel and separating objects from background. There are pros and cons of this method. It is absolutely helpful, for example picking rotten apples from processing line based on the different color of them. However, detecting color is not one hundred percent accuracy.

In Figure 3.9, the HSV filter mask is applied to the raw image that captured from Kinect. After filtering, these three objects are detected and covered by black edges. Further information of these objects (position, shape, spatial relation, etc.) is also collected and published to the knowledge base.

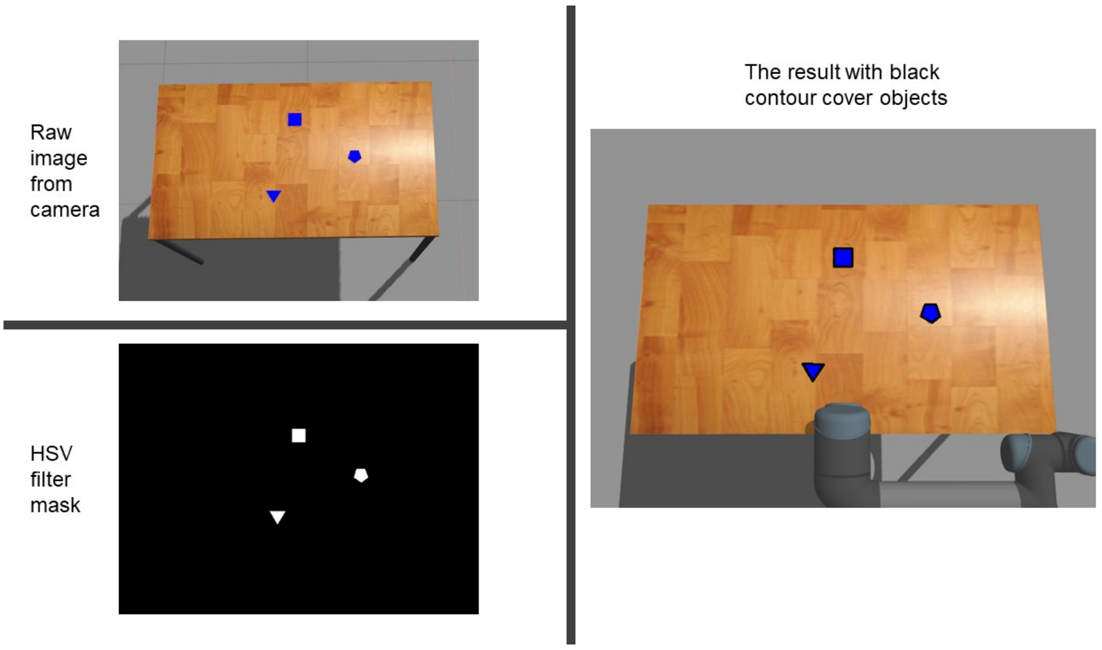


Figure 3.9. *HSV filter mask is applied for input image to segment the objects.*

3.2.3 Machine Learning - TensorFlow object detection

In next chapter, the alternative Visual Perceptor module is implemented in the system for experiment. The core detection unit in new module is TensorFlow object detection object [18].

TensorFlow is an open-source platform that support machine learning. Under the scope of this thesis, the Object Detection Library of TensorFlow is used for the alternative module. There are some pre-trained models included with this library and the SSD MobileNet COCO v1 dataset is chosen for this task. There are hundred classes representing the ordinary objects in the dataset. It also contains the feature extracted from thousand images of each class. To detect the object in current scene, the TensorFlow model creates multiple boxes and scan the input captured image or streaming image. At every position, it compare the probability of similar between the input data and the trained featured. User defined the threshold for accepting the detected objects. The threshold used in this thesis is 80 percent, which means if the detected object has equal or more than 80 percent similar to the considered feature, the class of that feature is labled for the detected object. The result of labeled object is overlaid the original image as a box with label and similarity covering the object. The example result is show in Figure 3.10. More details about TensorFlow in practical work will be introduced in next chapter.

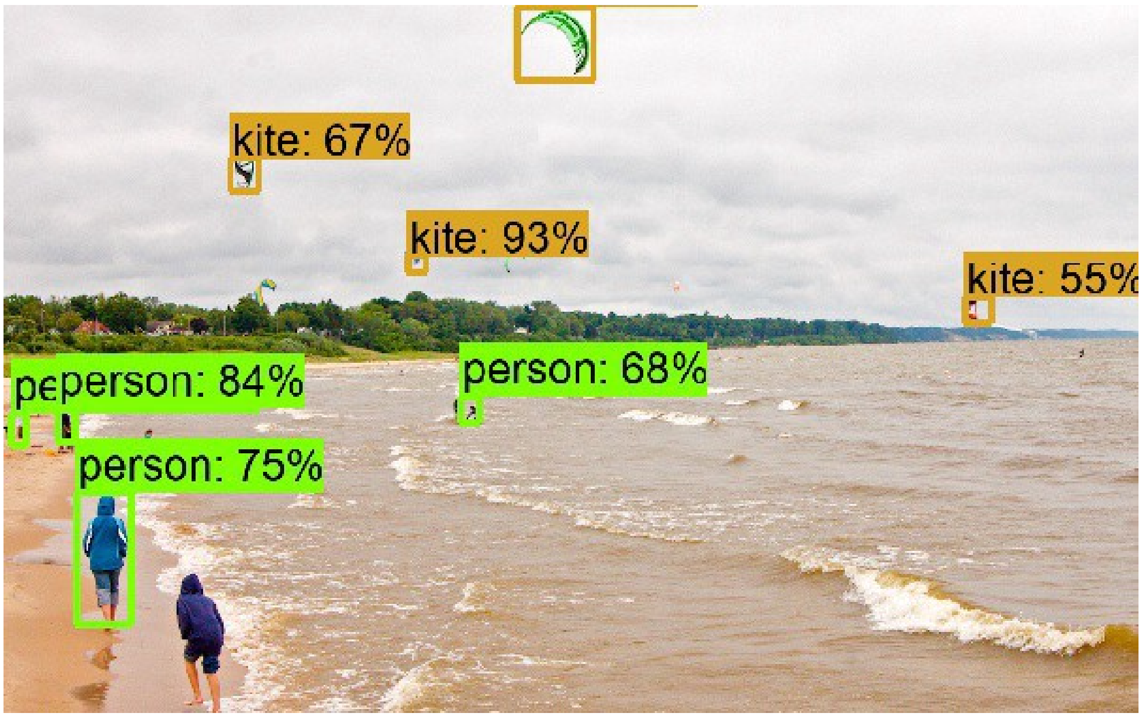


Figure 3.10. *TensorFlow object detection API example [18].*

3.3 ROS and Gazebo simulation

3.3.1 ROS - Robotic Operation System

Started from 2007, the Robotic Operation System or ROS has become the most popular robotic middleware in the world. It is supported by many world brand robotic company and that is the reason why it is chose to be used as the environment of action side, where handles all task related to the field devices. Besides, ROS communication is easy to follow. It mostly works around the topic server where ROS topics are the place that data from field devices is published to and get subscribed by other ROS nodes. The consequence of being the universal middleware is that many software packages are developed for making ROS compatible with other environments. In this thesis, web environment is the target that we want to connect with ROS. To do that, we used an official package called "rosbridge_suite" for connecting ROS nodes through websocket protocol. Also, one of those nodes is embedded in a web server, which control the user interface site.

To describe the *rosbridge_suite* role in this study, Figure 3.11 displays an experimental setup, which includes a Linux-based laptop that handles ROS environment and other Window-based laptop that handle augmented visual control in Unity 3D environment. These two laptops are running in different operation system and their programs work in different style. However, they are using same WiFi network, thus it is the infrastructure for *rosbridge_server*, which is a library in *rosbridge_suite*, to use. Through websocket protocol, the Window-based laptop only need to send connection request to Linux-based laptop once. When the pipeline is created between two laptops, they have ability to exchange data.

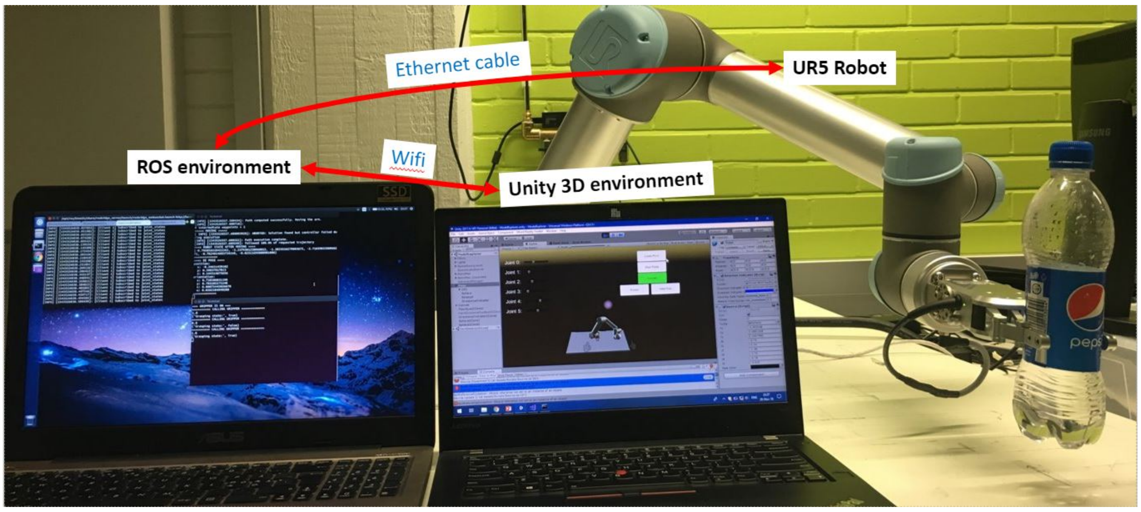


Figure 3.11. *rosbridge_suite setting up example*

Another problem to solve with this transmission is the difference in structure and type of data. Due to ROS using their own message structure, the user interface site, which is a webpage, should has ability to convert data to ROS message structure. Fortunately, the *rosbridge_suite* also has *rosbridge_library* package that has responsibility for handling JSON string and convert them to ROS message, and vice versa.

3.3.2 Gazebo environment

Gazebo is a simulation world that integrated into ROS. It is developed with physic engine and 3D graphic interface. For the purpose of this thesis, all models used in this thesis are simulated in Gazebo world (shown in Figure 3.12). UR5 manipulator is controlled by using MoveIt software. In details, the MoveIt software provides API that we could implement as a script for re-use calling. This solution is suitable because the purpose of this study is making emphasis of how flexible of the system with the knowledge-base ontology is.

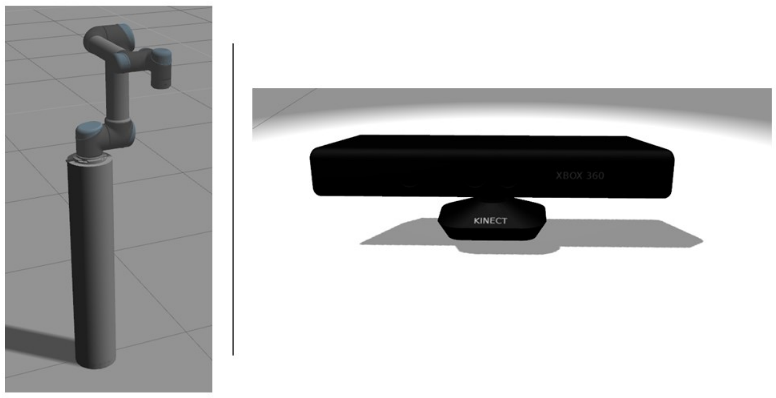


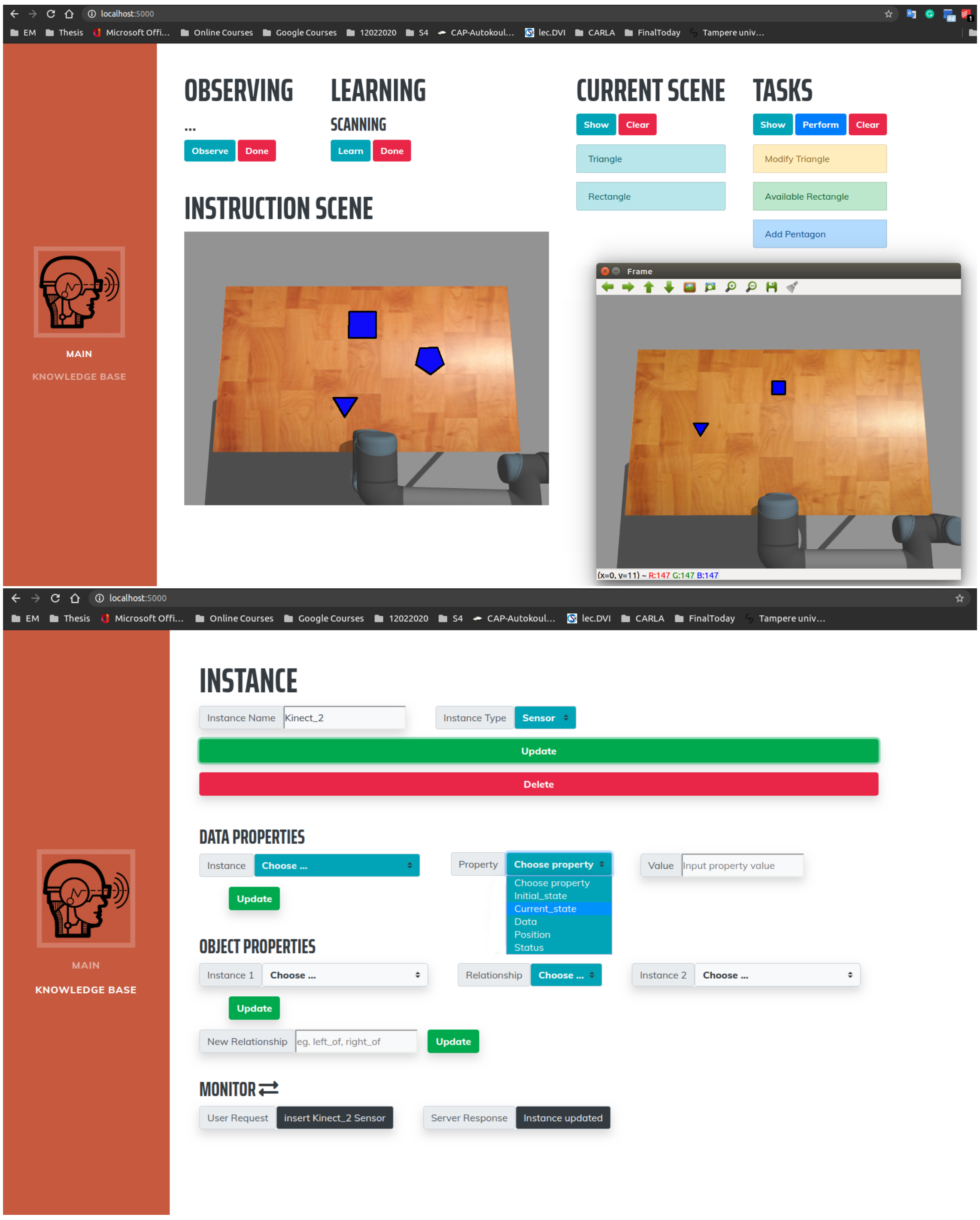
Figure 3.12. *UR5 and Kinect model in Gazebo environment.*

3.4 Web User Interface

Finally, the processing of the system is visualized on a webpage. To handle the interaction between user and the system, a web server is developed by Python-based Flask framework. Flask is considered instead of other website framework due to the lightness of processing and clear project structure. Since we want to integrate an knowledge-base ontology system that can be applied to any ROS related robotic devices, the simpler solution is chose.

In Figure 3.13, the webpage of user interface has two sites which are "MAIN" and "KNOWLEDGE BASE". At MAIN site, the user interface contains the options for user to control the manipulation system in ROS environment. The "Learning" tab, which contains "Learn" and "Done" buttons, is the section of initializing camera. By clicking on "Learn", the camera starts scanning the current scene of the table. Subsequently, the scanned data are transferred to the other unit for detecting objects in the scene and pointing out the differences between current scene and instruction scene. The instruction scene can be updated manually by the user and it is the desired scene that user want the system to complete. As showed in the top image of Figure 3.13, the tasks of the system are to "Modify Triangle" and to "Add Pentagon" as the Triangle is in wrong position and the Pentagon is missing in the current scene. Each task has their priority color for user recognition. It helps user easy to follow the process and what is needed to focus on.

In bottom image of Figure 3.13, the request and response are showed. This figure demonstrates an example of updating new Instance (Individual) in the knowledge base. A new sensor individual is created with name "Kinect_2". Beside updating new individual, other components of an ontology can be modified likewise as "Data Properties", "Object Properties" and the relations between individual.



Instance Name: Kinect_2
Instance Type: Sensor
User Request: insert Kinect_2 Sensor
Server Response: Instance updated

Figure 3.13. *The final version of user interface that has ability of displaying the progress (the top image) and updating the knowledge base (the bottom image). The interface of this webpage is built on Bootstrap version 4.*

4 EXPERIMENTS AND ANALYSIS

4.1 Experiments

To answer the research questions, the objectives listed in Chapter 1 need to be accomplished. The goal is to develop a flexibility framework so that it could be used in other system. The research questions are presented below.

- Is it possible to modulize the proposed framework?
- How can the user interact with the system from distance?

In order to provide the answers, we build up three experiments. To answer the first question, the system is designed as set of modules and each of them should be able to be replaced. The first experiment provides a view of the system when it uses different camera and different object detection method for testing if the system still works with alternative module. Equally, the second experiment evaluates how "Robot handler" block in Figure 3.1, which is developed with MoveIt software, control the real UR5 robot to prove that the system has ability to work with real robot. Besides, the second experiment also testing the flexibility when the system uses another robot instead of UR5. To tackle the second question, the user interface is designed as a webpage. This approach provides user the ability to interact remotely with the knowledge base, the robot and other devices in same network. The last experiment tests if the proposed system successfully works and shows the result of it under three different conditions.

In Chapter 3, all the preparation and necessary elements are introduced. The setting up of each experiment will be presented in next parts of this chapter.

4.1.1 Using Intel RealSense camera and Tensorflow object detection method

In Figure 3.1, the Server Side block and the Knowledge Base server is running on web base environment. This supports these units to work under different platform and are replaceable. However, at ROS environment side, there is still a question mark of doing such a thing. Particularly, in the ROS environment block, the "Robot handler" and its replaceable ability will be discussed in next section of "Remote control real UR5 arm with MoveIt software" and usually the "Gripper handler" is attached with manipulator as well.

Hence, this experiment focuses on the "Visual Perceptor" block, which requires a camera for input images and object detection method to seperate objects from the background. The default Gazebo Kinect camera and HSV filter will be replaced by RealSense camera and Tensorflow object detection.

Because Intel RealSense camera has unacceptable model in Gazebo simulation, we use real camera instead. The system now detects the real objects by Tensorflow object detection pre-trained model and control the independent UR5 Gazebo. The user interface is the developed for this experiment to reveal the detected objects and their spatial information.

When the system is running, user clicks on "Observe" button to start scanning the object on top of the table. The result from TensorFlow object detection model is showed in user interface and updated in ontology with high frequency because the processed image is from streaming line. To show the current objects, user clicks on the "Update" button and the "Objects" tab is revealed with a list. Clicking on the object in "Objects" list reveals more information tabs and "Actions" tab. By clicking the action in "Actions" tab, user commands the Gazebo UR5 to perform that action. Result of this experiment is showed in Figure 4.1.

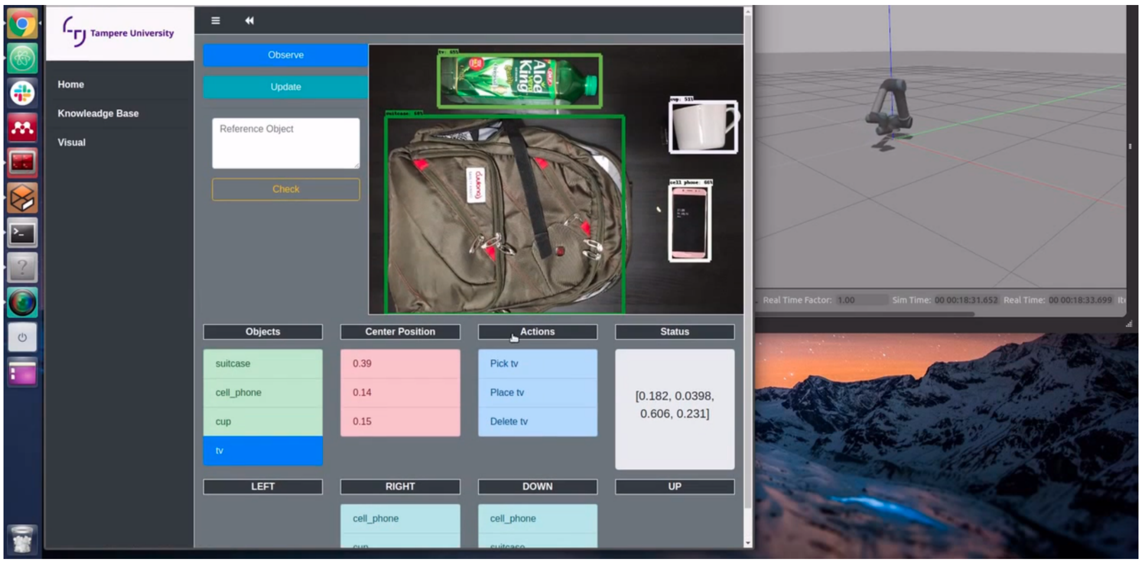


Figure 4.1. *Real RealSense camera and Tensorflow object detection are implemented in the system instead of Gazebo Kinect camera and HSV filter. The results of detection are updated in user interface for comfortably following.*

As can be seen in the figure, some objects have wrong labels. Specifically, in Figure 4.1 the bottle is labeled as a TV and the back bag is called suitcase. However, the method provides a good boundary of the objects, which in turns separate the objects from the background. The Gazebo UR5 moves to the position of object respect to estimated position of it in real world but Gazebo UR5 cannot perform picking or other task due to no object in Gazebo world. However, the main purpose of this experiment is testing the ability to work of the system with an alternative module and different sensor. Although module is changed, the experiments shows that the system still work.

4.1.2 Control real UR5 arm with MoveIt software

As described in Figure 3.11, a real UR5 robot was controlled by a ROS environment laptop. By connecting the laptop to the UR5 controller through network cable, the laptop receives data of the UR5 as joint movements, the current position of end-effector respected to robot base, etc. The user enters the desire coordinates in user interface to move the UR5's end-effector. These coordinates are respected to UR5 base also. The MoveIt interface plans the robot trajectory based on current state of the robot and the desire goal. If the goal is unreachable, the MoveIt interface sends a message of failing for user to adjust the goal. Otherwise, the MoveIt interface finishes planning with a "Successfully Planned" notification and executes the robot to move along planned trajectory. When the robot reaches the goal, a "Finished execution" message is updated for user. Requiring desire coordinates as input and noticing user the status of process are the good reason that MoveIt interface is chosen for "Robot handler" module in this proposed system. Additionally, the figure below displays the result that the real robot moving accordingly to the plan in ROS laptop with MoveIt interface. Markedly, the MoveIt interface to control the robot could be compacted in one script of coding.

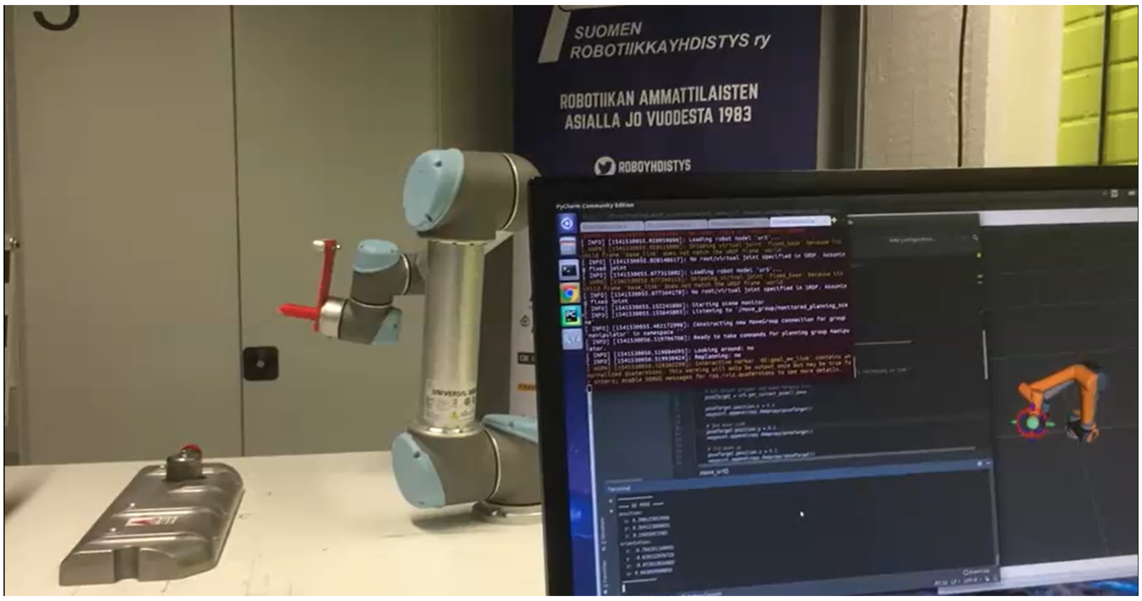


Figure 4.2. *Planning and executing real UR5 robot with MoveIt interface*

Besides, MoveIt provides the ability to define the robots in its structure. If the system uses a different manipulator, the new robot arm only needs to be defined with MoveIt Setting Up Assitance. With this result, the proposed system shows the ability that it could work with real UR5 robot.

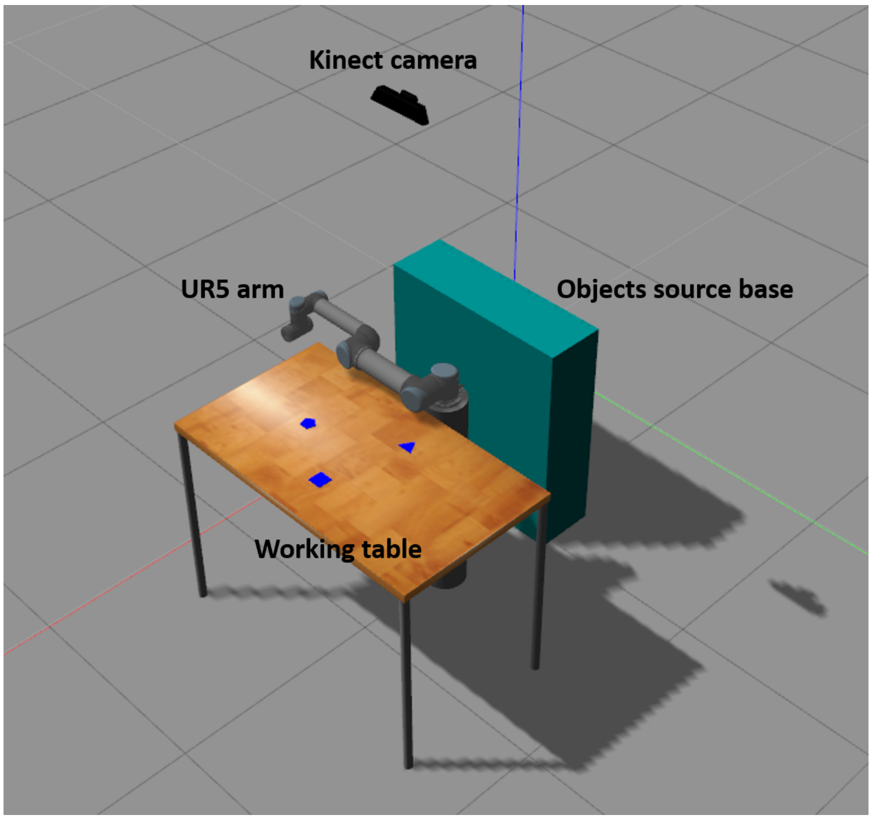


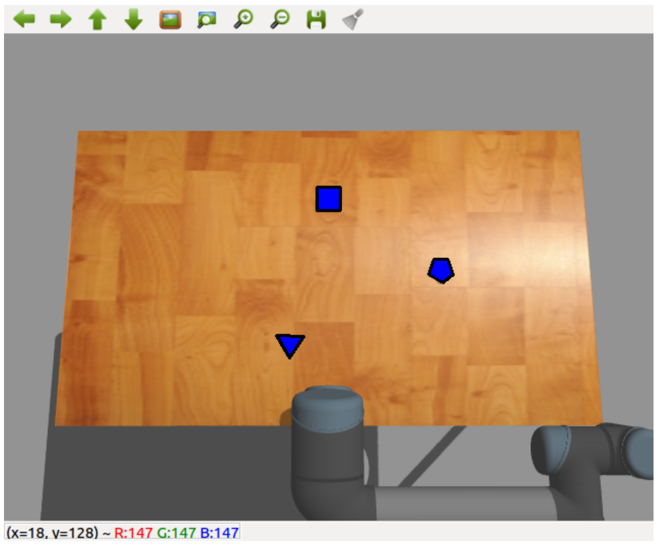
Figure 4.3. *The setting up in Gazebo environment*

4.1.3 UR5 in Gazebo completes the desire scene from random one

The initialization of this experiment is showed in Figure 4.3. The Kinect camera is responsible for detecting the differences between the current scene and the target scene, which is defined by user and published from user interface. Results from detecting part will be sent to the Reasoner for generating tasks that the robot should follow to complete the target scene. These tasks are revealed in user interface for confirmation. In case the tasks are not correctly, user cancels current tasks and requires the system to learn the scene again. If the user is satisfies with the tasks, one can activate the robot to follow these task by clicking "Perform" button as in Figure 3.13 top image.

To emphasis the adaptation of the system, three different target scenes were evaluated. With each target scene, the experiment will be repeated in ten times to demonstrate the repeatability of the system. These experiments starts by defining the target scene and updating it to the knowledge base. User arranges objects on the table after presses "Observe" button for starting the Kinect camera. When the user is satisfied with the arrangement, pressing "Done" button of "Observing" tab stops the scanning process of camera and updates extracted features as target scene information while the captured image is used to update instruction scene in user interface. After that, random scenes are created by rearranging the objects for testing. In the first two experiments, each one is carried out ten times each, thus 20 random scenes in total will be created. The last experiment will be evaluated with an empty input scene due to the limitation in computational power of the in-used laptop. The results is presented below.

Case 1: Three different objects



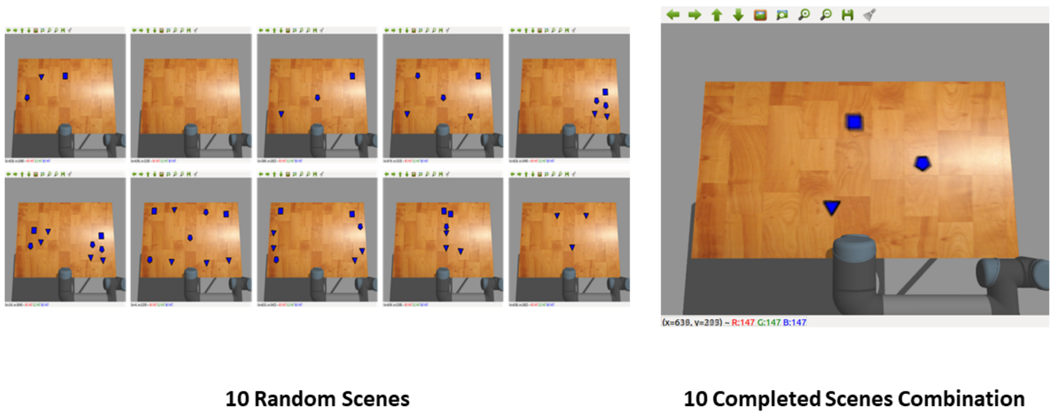
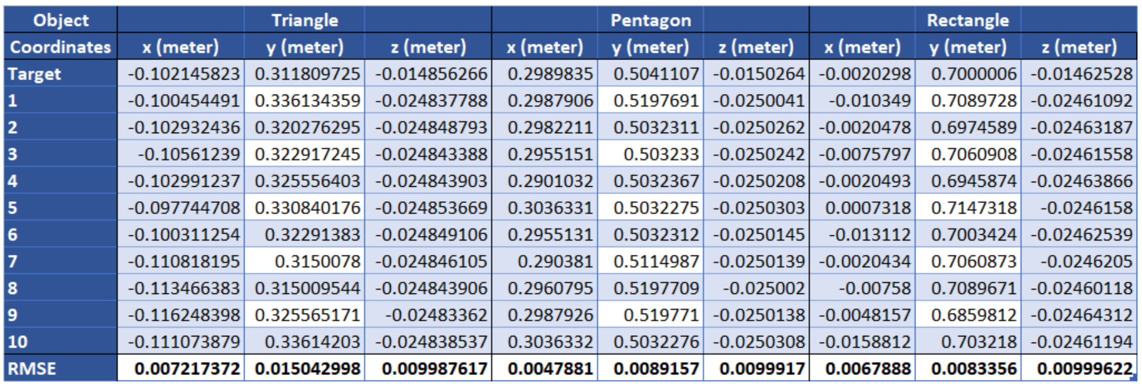

(x=638, y=288) ~ R:147 G:147 B:147
10 Random Scenes
10 Completed Scenes Combination

Figure 4.4. *Top image: the target scene of 3 different objects. Bottom image: the results from 10 times running system with random scenes.*

The result from this first experiment shows that the system complete the scene. The glow effect of objects from the completed scenes is acceptable. As can be seen, only Rectangle has blur edges while others have pretty clear edges. The main reason is that the UR5 manipulator has some small error offset between the measured position and the desired position. This can be accepted if the number of objects is small but it would cause problems in the case of multiple objects. The last case result will take this problem into account. The results are not different from the target scene.

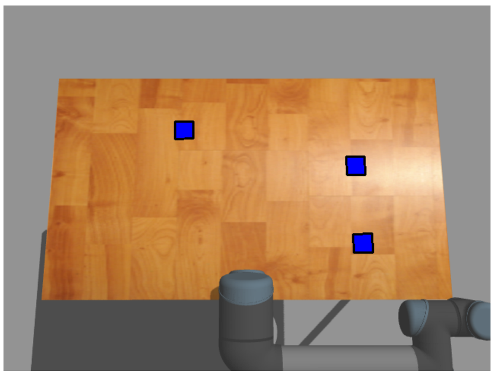


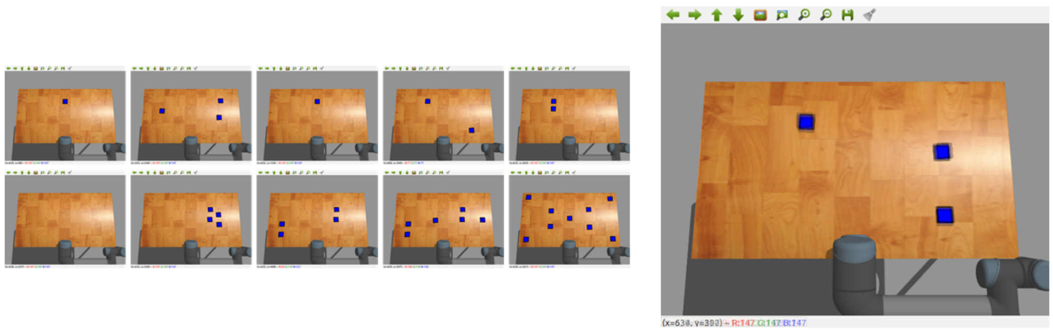
Object		Triangle			Pentagon			Rectangle	
Coordinates	x (meter)	y (meter)	z (meter)	x (meter)	y (meter)	z (meter)	x (meter)	y (meter)	z (meter)
Target	-0.102145823	0.311809725	-0.014856266	0.2989835	0.5041107	-0.0150264	-0.0020298	0.7000006	-0.01462528
1	-0.100454491	0.336134359	-0.024837788	0.2987906	0.5197691	-0.0250041	-0.010349	0.7089728	-0.02461092
2	-0.102932436	0.320276295	-0.024848793	0.2982211	0.5032311	-0.0250262	-0.0020478	0.6974589	-0.02463187
3	-0.10561239	0.322917245	-0.024843388	0.2955151	0.503233	-0.0250242	-0.0075797	0.7060908	-0.02461558
4	-0.102991237	0.325556403	-0.024843903	0.2901032	0.5032367	-0.0250208	-0.0020493	0.6945874	-0.02463866
5	-0.097744708	0.330840176	-0.024853669	0.3036331	0.5032275	-0.0250303	0.0007318	0.7147318	-0.0246158
6	-0.100311254	0.32291383	-0.024849106	0.2955131	0.5032312	-0.0250145	-0.013112	0.7003424	-0.02462539
7	-0.110818195	0.3150078	-0.024846105	0.290381	0.5114987	-0.0250139	-0.0020434	0.7060873	-0.0246205
8	-0.113466383	0.315009544	-0.024843906	0.2960795	0.5197709	-0.025002	-0.00758	0.7089671	-0.02460118
9	-0.116248398	0.325565171	-0.02483362	0.2987926	0.519771	-0.0250138	-0.0048157	0.6859812	-0.02464312
10	-0.111073879	0.33614203	-0.024838537	0.3036332	0.5032276	-0.0250308	-0.0158812	0.703218	-0.02461194
RMSE	**0.007217372**	**0.015042998**	**0.009987617**	**0.0047881**	**0.0089157**	**0.0099917**	**0.0067888**	**0.0083356**	**0.00999622**

Figure 4.5. *Experiment 1: Estimated position of each object in 10 attempts*

From Figure 4.5, the root mean square error is approximate 1 centimeter. The highest error is the y coordinate of Triangle with around 1.5 centimeter. There is small error in pose estimation due to 1 centimeter offset in every z coordinate. This calibration issue can be fixed by configuration the device. However, these results support that the proposed system has decent accuracy. Next case of experiment will test the precision of the system.

Case 2: Three similar objects



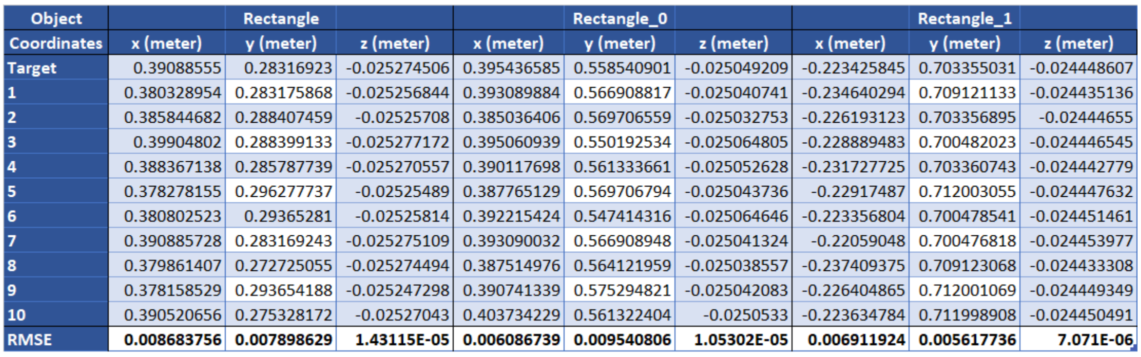


10 Random Scenes

10 Completed Scenes Combination

Figure 4.6. *Top image: the target scene of 3 similar objects. Bottom image: the results from 10 times running system with random scenes.*

In this experiment, the main purpose is to evaluate the precision of controlling robot arm. As can be seen in Figure 4.6, compared to the target scene, the Rectangles objects has glow effect on them. However, the scene is the same as the the target. For mode evidences, the data of this experiment are presented in 4.7.



Object		Rectangle			Rectangle_0			Rectangle_1	
Coordinates	x (meter)	y (meter)	z (meter)	x (meter)	y (meter)	z (meter)	x (meter)	y (meter)	z (meter)
Target	0.39088555	0.28316923	-0.025274506	0.395436585	0.558540901	-0.025049209	-0.223425845	0.703355031	-0.024448607
1	0.380328954	0.283175868	-0.025256844	0.393089884	0.566908817	-0.025040741	-0.234640294	0.709121133	-0.024435136
2	0.385844682	0.288407459	-0.02525708	0.385036406	0.569706559	-0.025032753	-0.226193123	0.703356895	-0.02444655
3	0.39904802	0.288399133	-0.025277172	0.395060939	0.550192534	-0.025064805	-0.228889483	0.700482023	-0.024446545
4	0.388367138	0.285787739	-0.025270557	0.390117698	0.561333661	-0.025052628	-0.231727725	0.703360743	-0.024442779
5	0.378278155	0.296277737	-0.02525489	0.387765129	0.569706794	-0.025043736	-0.22917487	0.712003055	-0.024447632
6	0.380802523	0.29365281	-0.02525814	0.392215424	0.547414316	-0.025064646	-0.223356804	0.700478541	-0.024451461
7	0.390885728	0.283169243	-0.025275109	0.393090032	0.566908948	-0.025041324	-0.22059048	0.700476818	-0.024453977
8	0.379861407	0.272725055	-0.025274494	0.387514976	0.564121959	-0.025038557	-0.237409375	0.709123068	-0.024433308
9	0.378158529	0.293654188	-0.025247298	0.390741339	0.575294821	-0.025042083	-0.226404865	0.712001069	-0.024449349
10	0.390520656	0.275328172	-0.02527043	0.403734229	0.561322404	-0.0250533	-0.223634784	0.711998908	-0.024450491
RMSE	**0.008683756**	**0.007898629**	**1.43115E-05**	**0.006086739**	**0.009540806**	**1.05302E-05**	**0.006911924**	**0.005617736**	**7.071E-06**

Figure 4.7. *Experiment 2: Estimated position of each object in 10 attempts*

From Figure 4.7, the root mean square error is less than 1 centimeter. The highest error is the y coordinate of Rectangle_0 with around 0.95 centimeter. These results proves that the proposed system has adequate precision when working with similar objects.

Case 3: Various objects in the scene

After testing the ability of handling a small number of objects, the system is evaluated with a crowded target scene, where the objects are positioned as "TUNI" word. There are 28 objects with different shapes that need to be placed at the right position to form the "TUNI" word. Unfortunately, the computational power of the laptop is not enough to handle too many objects, which in turns, causes frozen situation. Thus, this experiment starts with an empty scene instead of random scenes like in previous experiments.

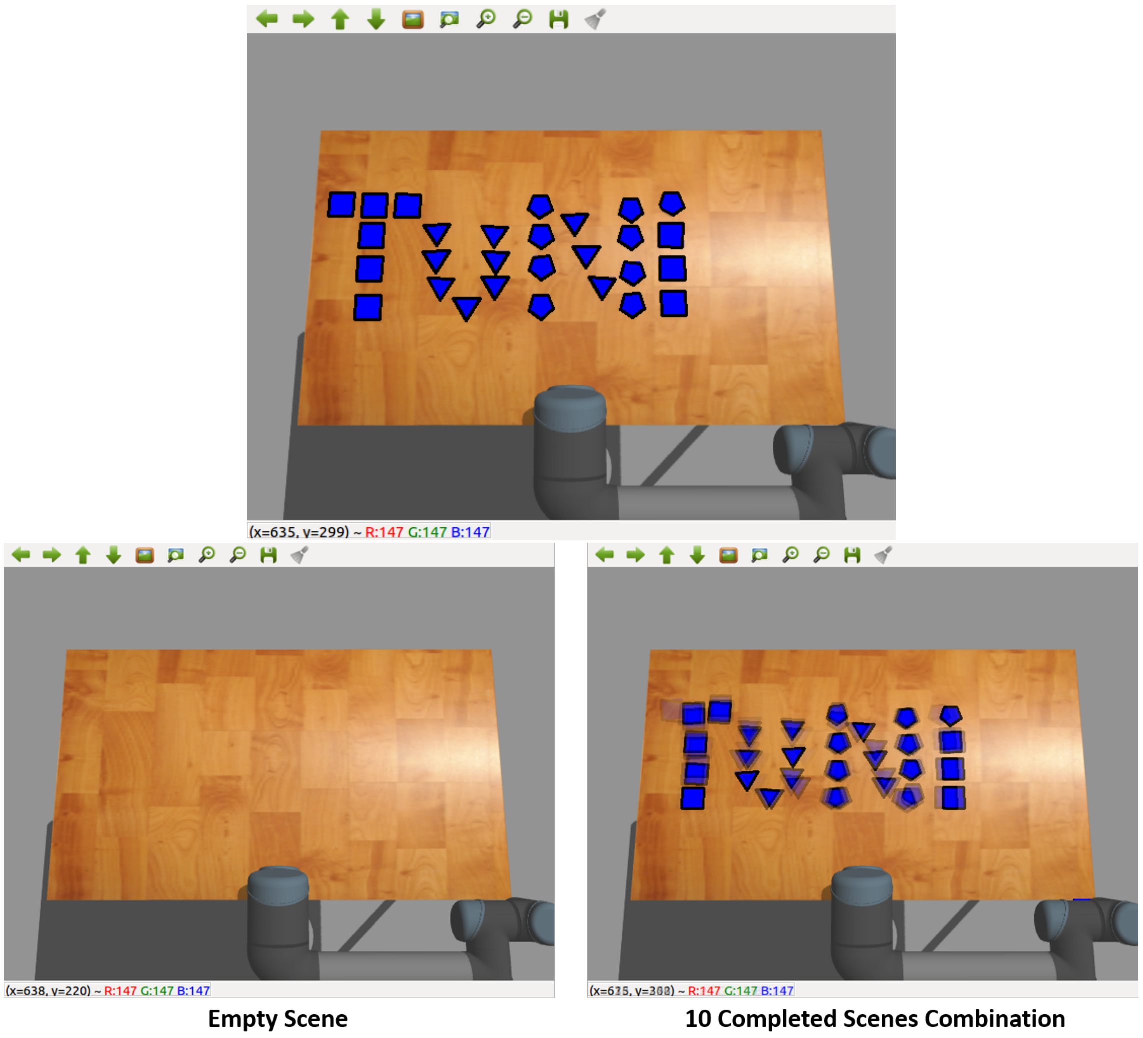

(x=635, y=299) ~ R:147 G:147 B:147
(x=638, y=220) ~ R:147 G:147 B:147
(x=615, y=308) ~ R:147 G:147 B:147
Empty Scene
10 Completed Scenes Combination

Figure 4.8. *Top image: the target scene of objects creating "TUNI" word. Bottom image: the results from 10 times running system.*

Although the most of objects are placed in acceptable position, the result from ten loops of running shows that the system have not complete the goal. There is no data table because of the system has not complete the scene. Due to the reaching ability of the UR5 in this setup, it cannot place the last piece of letter T on the left. Furthermore, the system produces not so high accuracy when handling many objects in a scene as there are big glow effect of the result.

Additional Testing: User interface in mobile device

In this testing, a new device is used to run the user interface. Because the user interface is basically a web page, it could be run from any device that supports web browser. The host computer, which runs the "Server Side" in Figure 3.1, and the smart phone connect to the same network. Because the Web UI server in this work is running with offline mode, the other devices that want to connect to the server should access through the IP address of the host computer. The result in Figure 4.9 shows that user can remote control the system by another device (The Web UI server only allows one thread at a time so only one user interface can be displayed). This result answer the second research question of how the system can be controlled from distance.

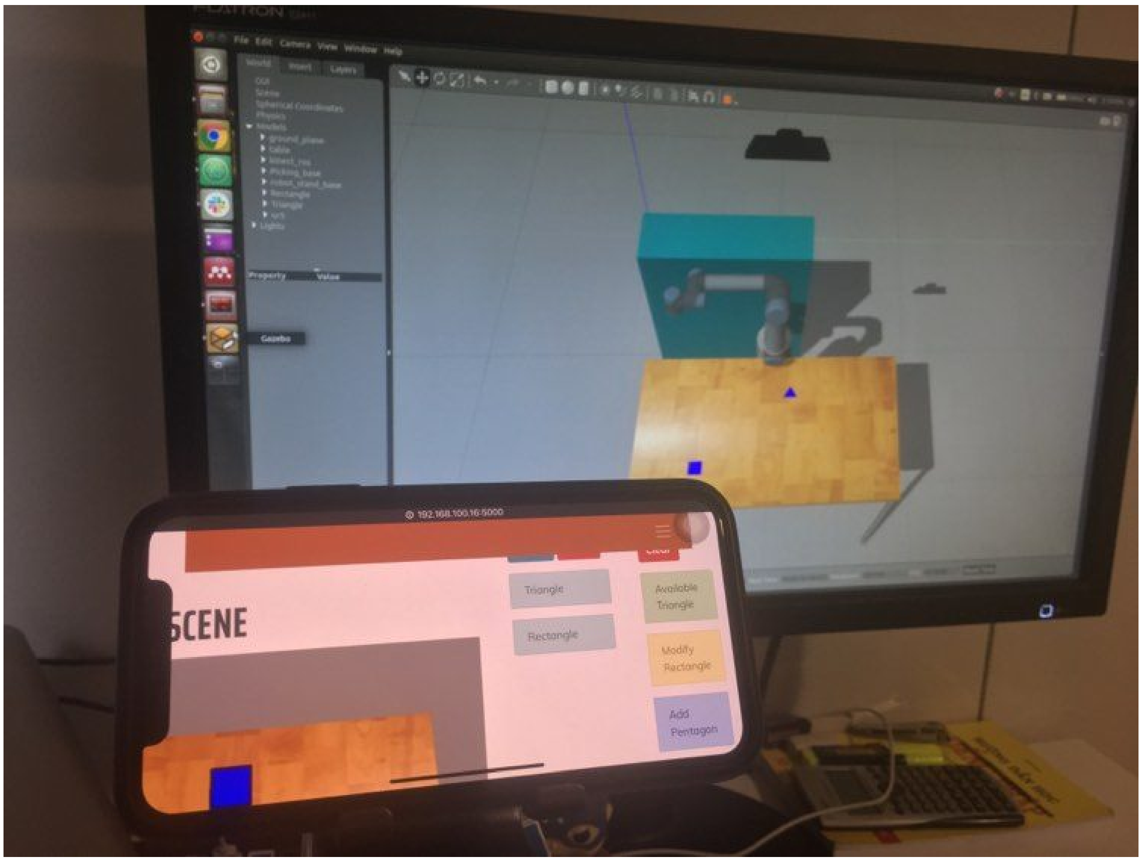


Figure 4.9. *The user interface is working on a smart phone. The user can remotely control the system by other device*

4.2 Discussion

The experiments show that the developed framework is capable of solving three mentioned questions

1. Are modules in the system (as Figure 3.1) replaceable?
2. Does the code that controls UR5 in simulation environment work with real robot

manipulator?

3. Is it possible to make the proposed system work with completing scene task?

The first experiment examines the Visual Perceptor modules by replacing with a new one. The result shows that the system can adapt the new module, which gives the evidence of the flexibility of the system. As mentioned in Chapter 3, the MoveIt script to control Gazebo UR5 in this thesis is implemented from a previous work with real UR5 robot. The result of the previous project is also approved [26] so it is true to use the proposed system of this thesis for real UR5 robot. The last experiment with three different conditions demonstrates that the system has ability to complete a scene from random scene. Moreover, it also proves that the proposed system has successfully developed from visual perception to simple semantic manipulation. It is possible for a robot to detect the difference between two scenarios and choose the actions to work with that difference for completing scene tasks. In additions, the connection between ROS platform and web environment is working well as the user interface can be run in a smart phone. To be applied in service field, the ability of distance control through Internet is a potential approach.

4.3 Limitations

Past through three experiments, there are some limitations of this thesis work that they need to be improved in future work. The drawbacks is listed below.

- HSV color detection can give wrong result if the background and the objects has similar color.
- Experiments are implemented in simulation world.
- There is no syncing between the reachable range of the robot and the threshold of detected object position.
- Only one camera is used.

5 CONCLUSION

From previous works, we learned that the majority of the research about semantic manipulation focus on improving the visual perception and grasping optimization rather than the application range. Semantic manipulation is usually discussed with the ability of understanding the scenario and reacting to the changes of surroundings. However, in the context of semantic manipulation, the visual detection and robot manipulation are closely associated, causing the manipulation system to work only in a specific environment. This raises a question if we can create a manipulation system that comprises independent modules, those share a common knowledge base. This opens an opportunity to expand to a semantic ecosystem where the robots communicate with other sensing elements to complete tasks under supervision of human from distance. Henceforth, this thesis aims to integrate a knowledge base into visual manipulation system in ROS environment. Furthermore, to increase the flexibility of the system, the visual detection and robot manipulation is compacted to independent modules.

The objectives of this thesis are to establish the connection between web and ROS environment at first, then developing a knowledge base manipulation system that has capable of completing a desire object pattern from a random scene and the proposed system has ability to substitute its modules. Accomplishing the mission leads to a bigger goal of semantic manipulation, where the robotic manipulator has capable of working safely in human-living environment.

This thesis was performed through three main chapters. Chapter 2 of theoretical background provide the key ideas and motivation for this topic. Chapter 3 discussed the research methodology that was used to solve the problems. Practical experiences are showed in Chapter 4 by means of experiments and results.

From theoretical background, the key definition of ontology and image processing are introduced. In this chapter, we focus on what is the ontology and how to develop an ontology so it can represent for a knowledge base that both human and robot manipulator could understand. Furthermore, the basic concept of image processing is mentioned in second part of Chapter 2.

The final system structure is explained in Chapter 3. As the system should be a group of independent modules, each module should work properly to make the whole system work. In addition, this chapter also describe the system architecture and its main modules. These modules are shown in the following list.

- Knowledge Base Server
- ROS environment
    1. Visual Perceptor: Gazebo Kinect camera configuration and HSV filter processing.
    2. Gripper Handler: Coding script control Gazebo Vacuum Gripper.
    3. Robot Handler: MoveIt-based interface for control the Gazebo UR5.
- Server side
    1. Reasoner: Query the knowledge base and extract reaction for UR5.
    2. Web UI Server: Handle the interaction of user interface with both user and ROS side modules.
    3. Ontology UI (optional): user interface of Apache Jena Fuseki for directly modifying the ontology.

The performance of proposed system is evaluated in Chapter 4. This chapter remarks the important questions this thesis want to tackle. To provide justified answers for those questions, three experiments are conducted. The first experiment attempts to prove that the system with the center of the knowledge base server could adapt new independent module. There are changes in Visual Perceptor module where real RealSense camera and Tensorflow object detection unit replace the default Gazebo Kinect and HSV filter processing. Besides, the interface of web UI is also modified to display information collected from new Visual Perceptor module. The second experiment is carried out in our previous project work at Tampere University. This experiment is proposed to provide evidence that the script control Gazebo UR5 in this thesis could control the real UR5 robot. In that work, the operator uses Hololens AR kit to send command to UR5 arm controller. This controller is actually a MoveIt interface, which receives the commands to plan and executes the UR5 manipulator. The result of previous work shows that MoveIt interface script can work robustly with the real UR5 arm. Last but not least, the final experiment evaluates the performance of the proposed system in three different cases. The experiment results show that, in first two cases, the proposed system completes the scene with convinced data. Albeit, there is inconvenience to complete the scene in the last case where the system cannot fully complete the scene. This inconvenience is a drawback of the proposed system and needs to be improved by future work. Overall, the system is able to communicate between two different platforms and complete the desired scenario from a random scene. The performance of proposed system is satisfactory.

5.1 Future Works

The development and implementation of this thesis work suggests the list below as possible future works for improving and expanding the proposed system.

- Implement with real devices.
- Expand the system with more robots and cameras sharing same knowledge base server.
- Improve Visual Perceptor module with better result.
- Sync the limitation of modules together for smoothly performance.
- Run the Server Side online so the user interface could be accessed through Internet.

The proposed system is evaluated in simulation and it provides potential to implement with practical devices. Although the system currently work in limited scene due to the reachability of the robot and one fixed camera, these limitations can be reduced by merging more cameras to expand the scenario to detect and by developing the collaboration between different robots that shares the common knowledge base. In conclusion, this work, including the suggested improvement tasks, will share a part of contribution towards semantic manipulation and lead to safety human-robot interaction.

REFERENCES

[1] E. E. Aksoy, A. Abramov, F. Wörgötter and B. Dellen. Categorizing object-action relations from semantic scene graphs. *2010 IEEE International Conference on Robotics and Automation*. 2010, 398–405.

[2] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano and M. Vincze. Multimodal cue integration through Hypotheses Verification for RGB-D object recognition and 6DOF pose estimation. May 2013, 2104–2111. ISBN: 978-1-4673-5641-1. DOI: 10.1109/ICRA.2013.6630859.

[3] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano and M. Vincze. Multimodal cue integration through Hypotheses Verification for RGB-D object recognition and 6DOF pose estimation. May 2013, 2104–2111. ISBN: 978-1-4673-5641-1. DOI: 10.1109/ICRA.2013.6630859.

[4] A. Angleraud, Q. Houbre, V. Kyrki and R. Pieters. Human-Robot Interactive Learning Architecture using Ontologies and Symbol Manipulation. Aug. 2018, 384–389. DOI: 10.1109/ROMAN.2018.8525580.

[5] A. Angleraud, Q. Houbre and R. Pieters. Teaching semantics and skills for human-robot collaboration. *Paladyn, Journal of Behavioral Robotics* 10 (Jan. 2019), 318–329. DOI: 10.1515/pjbr-2019-0025.

[6] J. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, M. Klingensmith, J. Libby, L. Yu, N. Pollard, M. Pivtoraiko, J. Valois and R. Zhu. An integrated system for autonomous robotics manipulation. Oct. 2012, 2955–2962. DOI: 10.1109/iros.2012.6385888.

[7] C. Bersch, D. Pangercic, S. Osentoski, K. Hausman, Z. Marton, R. Ueda, K. Okada and M. Beetz. Segmentation of Textured and Textureless Objects through Interactive Perception. *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments* (July 2012).

[8] BMW-Werk-Leipzig. *BMW plant in Leipzig, Germany: Spot welding of BMW 3 series car bodies with KUKA industrial robots*. 19 July 2005. URL: https://en.wikipedia.org/wiki/High_tech#/media/File:BMW_Leipzig_MEDIA_050719_Download_Karosseriebau_max.jpg.

[9] C. Chao, M. Cakmak and A. Thomaz. Towards grounding concepts for transfer in goal learning from demonstration. Vol. 2. Sept. 2011, 1–6. DOI: 10.1109/DEVLRN.2011.6037321.

[10] M. Ciocarlie, K. Hsiao, E. Jones, S. Chitta, R. Rusu and I. Ucan. Towards Reliable Grasping and Manipulation in Household Environments. *Springer Tracts in Advanced Robotics* 79 (Jan. 2010). DOI: 10.1007/978-3-642-28572-1_17.

[11] M. Ciocarlie, K. Hsiao, E. Jones, S. Chitta, R. Rusu and I. Ucan. Towards Reliable Grasping and Manipulation in Household Environments. *Springer Tracts in Advanced Robotics* 79 (Jan. 2010). DOI: 10.1007/978-3-642-28572-1_17.

[12] G. Della Penna, D. Magazzeni and S. Orefice. A formal framework to represent spatial knowledge. *Knowledge and Information Systems* 51 (Aug. 2016). DOI: 10.1007/s10115-016-0975-3.

[13] S. Ekvall and D. Kragic. Robot Learning from Demonstration: A Task-level Planning Approach. *International Journal of Advanced Robotic Systems* 5 (Sept. 2008). DOI: 10.5772/5611.

[14] R. Girshick, J. Donahue, T. Darrell and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. Nov. 2014.

[15] C. Gomez, A. Hernández, J. Crespo and R. Barber. A topological navigation system for indoor environments based on perception events. *International Journal of Advanced Robotic Systems* 14 (Feb. 2017), 172988141667813. DOI: 10.1177/1729881416678134.

[16] B. Horn. *Robot Vision*. Jan. 1986. ISBN: 978-0-262-08159-7.

[17] M. Horvath. *Substractive color mixing*. 26 October 2006. URL: https://commons.wikimedia.org/wiki/File:SubtractiveColor.svg.

[18] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR* abs/1611.10012 (2016). arXiv: 1611.10012. URL: http://arxiv.org/abs/1611.10012.

[19] S. Iqbal, J. Tremblay, T. To, J. Cheng, E. Leitch, A. Campbell, K. Leung, D. McKay and S. Birchfield. Directional Semantic Grasping of Real-World Objects: From Simulation to Reality. (Sept. 2019).

[20] jena.apache.org. *Jena architecture overview*. 2011. URL: https://jena.apache.org/about_jena/architecture.html.

[21] J. (Jia. A Machine Vision Application for Industrial Assembly Inspection. *Machine Vision, International Conference on* 0 (Jan. 2009), 172–176. DOI: 10.1109/ICMV.2009.51.

[22] H. Kasaei, M. Oliveira, G. H. Lim, L. Seabra Lopes and A. Tomé. Towards Lifelong Assistive Robotics: A Tight Coupling between Object Perception and Manipulation. *Neurocomputing* 291 (Feb. 2018). DOI: 10.1016/j.neucom.2018.02.066.

[23] R. Kittmann, T. Fröhlich, J. Schäfer, U. Reiser, F. Weisshardt and A. Haug. Let me Introduce Myself: I am Care-O-bot 4, a Gentleman Robot. Sept. 2015.

[24] O. Kroemer, S. Niekum and G. D. Konidaris. A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms. *CoRR* abs/1907.03146 (2019). arXiv: 1907.03146. URL: http://arxiv.org/abs/1907.03146.

[25] B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence* 119 (Apr. 2000), 191–233. DOI: 10.1016/S0004-3702(00)00017-5.

[26] T. N. Le, T. Le, T. Bui and P. Ganguly. Human Robot Interaction Using Augmented Reality (Hololens) on UR5. (Dec. 2018). URL: https://research.tuni.fi/robolabtampere/projects/.

[27] I. Lenz, H. Lee and A. Saxena. Deep Learning for Detecting Robotic Grasps. *International Journal of Robotics Research* 34 (Jan. 2013). DOI: 10.1177/0278364914549607.

[28] G. H. Lim, I. H. Suh and H. Suh. Ontology-Based Unified Robot Knowledge for Service Robots in Indoor Environments. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 41 (June 2011), 492–509. DOI: 10.1109/TSMCA.2010.2076404.

[29] X. Lin and T. Chen. A Qualitative Approach for the Elderly's Needs in Service Robots Design. *Proceedings of the 2018 International Conference on Service Robotics Technologies*. ICSRT '18. Chengdu, China: Association for Computing Machinery, 2018, 67–72. ISBN: 9781450364348. DOI: 10.1145/3208833.3208846. URL: https://doi.org/10.1145/3208833.3208846.

[30] Z. Liu, D. Chen, K. Wurm and G. Wichert. Table-top scene analysis using knowledge-supervised MCMC. *Robotics and Computer-Integrated Manufacturing* 33 (Sept. 2014). DOI: 10.1016/j.rcim.2014.08.009.

[31] R. Müller. Ontology in Automation. Jan. 2008.

[32] F. Natalya and L. M. Deborah. Ontology Development 101: A Guide to Creating Your First Ontology SMI. *Technical report SMI-2001-0880, Stanford University, 2001* (2001).

[33] B. Pitzer, S. Osentoski, G. Jay, C. Crick and O. Jenkins. PR2 Remote Lab: An environment for remote development and experimentation. *Proceedings - IEEE International Conference on Robotics and Automation* (May 2012), 3200–3205. DOI: 10.1109/ICRA.2012.6224653.

[34] C. Premebida, R. Ambrus and Z. Marton. Intelligent Robotic Perception Systems. Nov. 2018. DOI: 10.5772/intechopen.79742.

[35] A. Richtsfeld, T. Mörwald, J. Prankl, J. Balzer, M. Zillich and M. Vincze. Towards Scene Understanding – Object Segmentation Using RGBD-Images. (May 2020).

[36] R. Rusu. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz* 24 (Nov. 2010). DOI: 10.1007/s13218-010-0059-6.

[37] R. Rusu, Z. Marton, N. Blodow, A. Holzbach and M. Beetz. Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments. Dec. 2009, 3601–3608. DOI: 10.1109/IROS.2009.5354759.

[38] R. Rusu, I. Sucan, B. Gerkey, S. Chitta, M. Beetz and L. Kavraki. Real-time Perception-Guided Motion Planning for a Personal Robot. Dec. 2009, 4245–4252. DOI: 10.1109/IROS.2009.5354396.

[39] T. R. Savarimuthu, A. G. Buch, C. Schlette, N. Wantia, J. Roßmann, D. Martínez, G. Alenyà, C. Torras, A. Ude, B. Nemec, A. Kramberger, F. Wörgötter, E. E. Aksoy, J. Papon, S. Haller, J. Piater and N. Krüger. Teaching a Robot the Semantics of

Assembly Tasks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48.5 (2018), 670–692.

[40] M. Schwarz, A. Milan, A. S. Periyasamy and S. Behnke. RGB-D Object Detection and Semantic Segmentation for Autonomous Manipulation in Clutter. *International Journal of Robotics Research (IJRR)* (June 2017). DOI: 10.1177/0278364917713117.

[41] B. Siciliano, L. Sciavicco, V. Luigi and G. Oriolo. *Robotics: Modelling, Planning and Control*. Jan. 2011. DOI: 10.1007/978-1-84628-642-1.

[42] Z. Sui, O. Jenkins and K. Desingh. Axiomatic particle filtering for goal-directed robotic manipulation. Sept. 2015, 4429–4436. DOI: 10.1109/IROS.2015.7354006.

[43] Z. Sui, L. Xiang, O. Jenkins and K. Desingh. Goal-directed robot manipulation through axiomatic scene estimation. *The International Journal of Robotics Research* 36 (Jan. 2017), 86–104. DOI: 10.1177/0278364916683444.

[44] G. Taylor and L. Kleeman. *Visual Perception and Robotic Manipulation - 3D Object Recognition, Tracking and Hand-Eye Coordination*. Jan. 2006. ISBN: 978-3-540-33454-5. DOI: 10.1007/11540151.

[45] W. Thompson, R. Fleming, S. Creem-Regehr and J. K. Stefanucci. *Visual Perception from a Computer Graphics Perspective*. 1st. USA: A. K. Peters, Ltd., 2011. ISBN: 1568814658.

[46] E. Tosello, Z. Fan and E. Pagello. A Semantic Knowledge Base for Cognitive Robotics Manipulation. Nov. 2015.

[47] Unknown. *Ontology-dimensions-map*. 18 November 2013. URL: https://wiki.opensemanticframework.org/index.php/File:Ontology-dimensions-map.png.

[48] H. Veeraraghavan and M. Veloso. Teaching sequential tasks with repetition through demonstration. Vol. 3. Jan. 2008, 1357–1360. DOI: 10.1145/1402821.1402871.

[49] w3c.org. *SPARQL*. 2009. URL: https://www.w3.org/wiki/SPARQL.

[50] T. Watanabe, K. Yamazaki and Y. Yokokohji. Survey of robotic manipulation studies intending practical applications in real environments -object recognition, soft robot hand, and challenge program and benchmarking-. *Advanced Robotics* 31.19-20 (2017), 1114–1132. DOI: 10.1080/01691864.2017.1365010. eprint: https://doi.org/10.1080/01691864.2017.1365010. URL: https://doi.org/10.1080/01691864.2017.1365010.

[51] Y. Yang, Y. Li, C. Fermüller and Y. Aloimonos. Robot Learning Manipulation Action Plans by " Watching " Unconstrained Videos from the World Wide Web. Jan. 2015.

[52] Z. Yingshen, P. Fillatreau, H. Karray and B. Archimède. An Ontology-Based Approach Towards Coupling Task and Path Planning for the Simulation of Manipulation Tasks. Oct. 2018. DOI: 10.1109/AICCSA.2018.8612805.

[53] Yu Du, C. W. de Silva and Dong Liu. A multi-agent hybrid cognitive architecture with self-awareness for homecare robot. *2014 9th International Conference on Computer Science Education*. 2014, 223–228.

[54] Z. Zeng, Y. Zhou, O. C. Jenkins and K. Desingh. Semantic Mapping with Simultaneous Object Detection and Localization. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 911–918.

[55] Z. Zeng, Z. Zhou, Z. Sui and O. Jenkins. Semantic Robot Programming for Goal-Directed Manipulation in Cluttered Scenes. May 2018, 7462–7469. DOI: `10.1109/ICRA.2018.8460538`.

[56] Y. Zhang, L. Li, M. Ripperger, J. Nicho, M. Veeraraghavan and A. Fumagalli. Gilbreth: A Conveyor-Belt Based Pick-and-Sort Industrial Robotics Application. *2018 Second IEEE International Conference on Robotic Computing (IRC)*. 2018, 17–24.