

TIDY UP FUNCTION DEVELOPMENT FOR SERVICE ROBOT

By

Liew Zhao Ying

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2019

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____

Academic Session: _____

I _____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:

Supervisor's name

Date: _____

Date: _____

TIDY UP FUNCTION DEVELOPMENT FOR SERVICE ROBOT

By

Liew Zhao Ying

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2019

DECLARATION OF ORIGINALITY

I declare that this report entitled “**TIDY UP FUNCTION DEVELOPMENT FOR SERVICE ROBOT**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Mr Teoh Shen Khang, who has given me this bright opportunity to involve in a robot vision project. It is my first step to establish a career in machine vision field. I am extremely thankful to him for his support and guidance.

To a very special person in my life, Ong Chi Yang, for his patience, unconditional support and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACT

This project develops an indoor service robot with tidy up function. It will provide an affordable solution for home maintenance, especially for busy working adults that have children. Regular home organisation will be made possible with minimal effort. The robot used in this project is Turtlebot3 Burger for its affordability. It is equipped with a custom-made robot arm for manipulation purpose and an RGB-D camera for object detection. The open source ROS and OpenCV are used as the main software framework in this project. Object detection is performed by combining image pre-processing techniques and Haar cascade for a machine learning based approach. Depth images are used to calculate object distance from the robot for robot arm control and robot motion planning. The resulting trained object detector can correctly identify toys in a scene consisting different types of small objects. The robot can update itself to detect and track the object if the object's location has changed. However, when objects of interests are placed close to each other or close to the camera, the object detector can give false positive results or fail to detect the object due to limitation of the LBP cascade algorithm and depth sensor.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT.....	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	x
LIST OF ABBREVIATIONS	xi
Chapter 1 Introduction.....	1
1.1 Problem Statement and Motivation.....	1
1.2 Project Scope	2
1.3 Project Objectives	2
1.4 Impact, Significance and Contribution	3
1.5 Background Information.....	4
1.5.1 Robotics in General.....	4
1.5.2 Household Robots and Cloud Robots	4
1.5.3 ROS	5
1.5.4 Machine Learning.....	5
1.5.5 Computer Vision	6
1.6 What have been achieved	6
1.7 Report Organization	6
Chapter 2 Literature Review	7
2.1 Literature Review.....	7
2.1.1 McBot.....	7
2.1.2 Autrobot.....	10
2.1.3 HERB	11
2.1.4 PR2	12
2.1.5 Johnny.....	14
2.2 Critical Remarks of Previous Works	15
Chapter 3 System Design.....	16
3.1 Methodology	16

3.2	Tools to Use.....	17
3.2.1	TurtleBot 3 Burger	17
3.2.2	Robot Arm	18
3.2.3	RGB-D Camera.....	19
3.2.4	ROS	19
3.2.5	OpenCV	20
3.2.6	Arduino IDE.....	20
Chapter 4	Project Implementation.....	21
4.1	System Design	21
4.1.1	Hardware Block Diagram	21
4.1.2	System Flowchart.....	22
4.2	ROS Setup.....	23
4.2.1	ROS Setup on Remote PC.....	23
4.2.2	ROS Setup on TurtleBot3 Burger	24
4.2.3	Network Configuration	24
4.3	Hardware Interfacing and Setup	25
4.3.1	OpenCR Setup.....	26
4.3.2	Bring Up TurtleBot.....	30
4.4	Custom Object Detector	32
4.5	Main Program	33
4.5.1	Image Topic Subscription	33
4.5.2	Image Pre-processing.....	34
4.5.3	Object Detection in Input Image	35
4.5.4	Turning Robot to align midpoint to object centroid.....	37
4.5.5	Pick Up Object at Object Location	39
4.5.6	Returning to Base.....	39
4.5.7	Placing Object at Base and Returning to Search for Toy Object.....	40
4.6	Visualisation of ROS Computational Graph.....	41
4.7	Implementation Issues and Challenges	42
Chapter 5	System Testing.....	44
5.1	Test Scenario: Tidy Up Area with Single Toy Object	44
5.2	Test Result.....	45

Chapter 6 Conclusion	50
6.1 Project Review	50
6.2 Future Work	50
Bibliography	52
Appendix A: Datasheets	A-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 1-1	Roomba® 980 vacuuming robot.	2
Figure 2-1	System configuration of McBot.	7
Figure 2-2	Task processing scenario of McBot.	8
Figure 2-3	Floor equipped by RFID tags.	9
Figure 2-4	Autrebot	10
Figure 2-5	HERB	11
Figure 2-6	Tidy up experiment with PR2.	13
Figure 2-7	Johnny	14
Figure 3-1	Methodology.	16
Figure 3-2	Components of TurtleBot3 Burger.	17
Figure 3-3	Robot Arm mounted on TurtleBot Burger.	19
Figure 3-4	CREATIVE Sens3D	19
Figure 4-1	Hardware block diagram.	21
Figure 4-2	System Flowchart.	22
Figure 4-3	Install ROS on Remote PC.	23
Figure 4-4	Install Dependent ROS Packages.	23
Figure 4-5	Catkin make Dependent ROS Packages.	23
Figure 4-6	SSH to RPi.	24
Figure 4-7	Implemented System Front View (left) and Side View (right).	25
Figure 4-8	OpenCR Pin Map.	26
Figure 4-9	Allow USB port access and 32-bit compiler.	27
Figure 4-10	Adding subscriber node in turtlebot3_core.ino	27
Figure 4-11	Robot arm control function in turtlebot3_core.ino	28
Figure 4-12	Function callback in turtlebot3_core_config.h	29
Figure 4-13	Upload firmware to OpenCR.	29
Figure 4-14	Successful firmware upload to OpenCR.	30

Figure 4-15	Softkinetic camera launch files location.	31
Figure 4-16	One of the positive samples	32
Figure 4-17	One of the negative samples.	32
Figure 4-18	Generate text file to index all negative samples.	33
Figure 4-19	Generate info.lst by merging images.	33
Figure 4-20	Start LBP cascade training.	33
Figure 4-21	Subscribing to topics published by Softkinetic camera.	34
Figure 4-22	Performing image pre-processing.	34
Figure 4-23	Resulting resized image after colour masking.	35
Figure 4-24	Object detector.	36
Figure 4-25	Object detection returning first detected result.	36
Figure 4-26	Turning TurtleBot sideways.	37
Figure 4-27	Object detected has slight offset to right.	37
Figure 4-28	Object detected has slight offset to left.	38
Figure 4-29	Object detected at middle and its depth profile.	38
Figure 4-30	Depth data output.	38
Figure 4-31	Going forward to pick up object.	39
Figure 4-32	Returning to base.	39
Figure 4-33	Placing object and continue search.	40
Figure 4-34	rqt_graph of system.	41
Figure 5-1	Initial point for tidy up task.	44
Figure 5-2	Robot repositioned to face toy object.	45
Figure 5-3	Robot moving forward until toy object enters arm pick-up range.	45
Figure 5-4	Rotated toy object due to limited gripper opening.	46
Figure 5-5	Small bounding box issue.	46
Figure 5-6	Successful attempt on picking up toy.	47
Figure 5-7	Placing toy objects to initial point	47
Figure 5-8	False positive detection.	48

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Comparison of existing service robot with tidy up functionality.	16

LIST OF ABBREVIATIONS

<i>GATMO</i>	Generalized Approach to Tracking Movable Objects
<i>ICE</i>	Internet Communications Engine
<i>IDE</i>	Integrated Development Environment
<i>LBP</i>	Local Binary Pattern
<i>LTS</i>	Long Term Support
<i>MANO</i>	Mobile Autonomous Nomadic
<i>OpenCV</i>	Open Source Computer Vision Library
<i>PC</i>	Personal Computer
<i>RFID</i>	Radio Frequency Identification
<i>RGB-D</i>	Red Green Blue Depth
<i>ROS</i>	Robot Operating System
<i>RPi</i>	Raspberry Pi
<i>SD</i>	Secure Digital
<i>SDK</i>	Software Development Kit
<i>SLAM</i>	Simultaneous Localisation and Mapping
<i>YOLO</i>	You Only Look Once

Chapter 1 Introduction

1.1 Problem Statement and Motivation

According to Fausset et al. (2011), ageing adults will face difficulties when it comes to home maintenance. The elderly might risk injuring themselves if they were to complete household chores, same applies to the disabled. With more working adults in the society, people will find it harder to make time for chores. Families with children will find it troublesome to tidy up children's toys. Besides that, without regular maintenance to keep the household clean and organised, it can be hard for the owner to keep track of their belongings. People must decide whether to sacrifice their off time to maintain their household cleanliness or hire a maid to keep their house clean and tidy. In the long run, these solutions will be costly in terms of time, energy, and money. However, with the growing trend of automation systems, robotics can be applied to replace human to complete tasks. When compared to humans, robots are more durable, productive, effective, and consistent when it comes to executing physical tasks. Robots can perform the given task well without having to prioritise its own safety and they are very flexible, making it possible for robots to work under most environments.

Intelligent home service robots can navigate the household environment and occasionally, move objects by themselves. One well-known example of home service robot is the floor-cleaning robot Roomba (shown in Figure 1-1), which can vacuum debris or dust from the floor while it navigates around the household area autonomously. It has gained popularity due to its simplistic design, ability to carry out task autonomously and its affordability. However, there is limitation to the Roomba's ability to help in household chores due to lack of manipulating ability to tidy up house objects. To ensure good productivity of home service robots, there are several factors to be concerned. The robot needs to have good mobility in household surroundings to access constricted spaces. The robot should be intelligent when it needs to recognise messy places and avoid hazardous objects. It requires decent battery life to be able to operate for a few hours. The human-robot interaction should be easy and friendly for the users. Therefore, in this project we

would like to develop an affordable home service robot that can tidy up household autonomously and has a manipulator to interact with household objects.



Figure 1-1 Roomba® 980 vacuuming robot.

1.2 Project Scope

This project develops a home service robot that can perform tidy up task in a workspace by arranging disordered objects according to their category. The home service robot will be interacting with small household objects that humans normally interact with, such as stationery and electronic devices. This project integrates computer vision into automation task like object detection and object tracking. Finally, the task will be performed autonomously with minimal amount of user intervention.

1.3 Project Objectives

In general, this project aims to replace human resource for household chores that involves tidying up messy places. The product will be able to perform tidy up function without excessively relying on human input. The service robot will autonomously keep household space well organised by detecting toy objects and return them back to a goal set by the user such as a box. This project enables the user to easily keep track of household objects.

This project will mostly focus on computer vision and machine learning in order to accomplish the tidy up task. Computer vision is mainly used for object detection and localization, and mapping of the environment to plan for robot navigation path. Machine learning is implemented in the robot to allow smart decision making when the robot needs to categorise the object and justify the correct location to place the object. The robot should be able to determine the path to reach the target and head to the goal for object placement. Furthermore, the project emphasises on building a mobile service robot that will use the least possible amount of wired connection between the command issuing machine and the service robot through cloud-based service nodes. To be able to interact with objects, this project will focus on tuning the manipulator on the robot for natural grasp and release action for different object types.

However, this project will not cover areas such as mobile application to command the service robot due to time constraints. The project also does not cover interaction with larger objects. This is due to the size of the robot arm is miniature, thus the opening of the gripper is small, and payload limit of the service robot.

1.4 Impact, Significance and Contribution

The innovation of this project is to build an affordable service robot compared to other service robots available on the market that can group messy items orderly and place them back to the location set by the user. This will help to solve the home maintenance problem with minimal amount of attention required from the user. The service robot will act as an organizer in the household, reducing the owner's workload at home while keeping the owner's preference of organizing objects.

The TurtleBot3 is using the ROS, a meta-operating system that is open source. The ROS is well equipped with a wide variety of packages that is compatible with most hardware. This also means the project is highly expandable in terms of hardware by simply installing new hardware and controlling them with the open source packages found on the

ROS website. Hence, addition of new functionalities is possible by further utilising existing hardware available on the robot or by adding new hardware. Therefore, the service robot we developed has more flexibility for expansion and is more customizable according to the task requirement when compared to existing home service robots available on the market, at the same time keeping the price affordable.

1.5 Background Information

1.5.1 Robotics in General

Since the evolution of robotics technology, automation system is getting more widespread due to reduced cost with the availability of 3D printing and ease of integration between computer software and robotics. Other than that, the increased accessibility of man resource for designing and maintaining the robotic automation systems contributed to its growth. Depending on the application, different robots with a variety of characteristics are needed for the automation system. For example, industrial robots are used to boost productivity in the industry and reduce safety concerns due to their sturdy structure. Mobile robots are capable to move around freely without being constrained in one physical location, thus they are most likely to be used in rescue missions to navigate spaces that are too small for humans to fit in. Hence, depending on the task requirement, robots will have different specifications.

1.5.2 Household Robots and Cloud Robots

According to Kopacek (2016), household robot's purpose is to service people at their home. Household robots can be categorised into 2 types, indoor and outdoor. For example, indoor household robot includes vacuum cleaner robot whereas the lawn mower robot represents outdoor household robot. On the other hand, cloud robots perform cloud-based computing for tasks that requires intensive data processing, such as image processing, voice recognition and 3D mapping. This can benefit the robot since hardware scaling can be done by utilising other machine's computing power and it is not necessary to equip the robot with powerful computing device. Besides, using cloud computing can

lower the power consumption of the robot, thus the size of its portable power source can be reduced. Usually, these cloud robots would be communicating with the cloud computing device via ROS, which is a common system for robot communication. By combining 2 of these robot types, we can build a household service robot that can utilise cloud computing and perform complicated chores in the household.

1.5.3 ROS

The ROS is a Linux-based, open source, meta-operating system for robots, it handles the processes between applications and the computing resources (Pyo et al., 2017). ROS maintains the software and hardware's compatibility and functionality despite being developed from different sources. Message communication in robots is performed via publishing and subscribing to a topic. To start the process, a master is required to establish initial node connections. A node is the smallest unit of process in ROS that works like a single purpose program. In order to establish message communication, the publisher nodes need to publish message on a topic, such as keyboard inputs to control the robot; and the message will be received by subscriber nodes that subscribed to the same topic.

1.5.4 Machine Learning

To become an intelligent machine, artificial intelligence techniques are applied to train the robot to complete its job without relying on a full program for possible outcomes of an event. This provides countless opportunities for robots to learn new skill sets by observing human behaviour and improve existing functions. Machine learning is a subset of the artificial intelligence field, where machine learning algorithms are utilised to train the machine to learn automatically by accessing existing data in its system and come up with a solution for the problem.

1.5.5 Computer Vision

Computer vision is a way for machines to perceive and represent the world. The process starts with image acquisition from external sensors such as cameras. Next, the image data will be processed with algorithms to extract the necessary details for the task. For example, an image of a basketball can be interpreted as an object that has a round shape and a rough surface with consistent pattern, hence its basic features are extracted from the image for categorisation. Some sensors even provide the option to output 3D image data for more accurate result. The applications of computer vision include autonomous driving for self-driving cars, object recognition, face detection, and augmented reality.

1.6 What have been achieved

Previous work of the project includes integration of robot arm to the TurtleBot. The process also includes camera setup, weight balancing for robot when the robot arm is moving, and adjustments for robot arm's degree of freedom to pick up and drop object actions. Besides, collection of toy image datasets for object detector model have been carried out to provide a solution to detect instances of objects of toy class in video frames. LBP cascade is used in this project to provide faster detection speed.

1.7 Report Organization

The following chapters discusses the details of the project development. Chapter 2 provides literature review of related project, summary of the pros and cons of the projects and our proposed solution. Then, Chapter 3 contains system design details, including methodology and tools used in the project. In addition, Chapter 4 describes the project implementation details, whereas Chapter 5 delivers testing result and analysis. Lastly, Chapter 6 concludes the project and provides future improvements.

Chapter 2 Literature Review

2.1 Literature Review

Over the years, many researchers have been trying to develop solutions to solve the problem of maintaining household tidiness. Hence, there are various types of approach to implement automation into service robots. We will discuss similar products for the home service robot purpose and other robots that utilises similar technology with our proposed solution.

2.1.1 McBot

The Mess-Clean-up Robot (McBot) developed by Ma et al. (2008). The McBot has navigation and manipulation system to clean up messy places. For McBot's navigation system, it is using RFID and gyro sensors, McBot can achieve self-localisation. For object manipulation, 2 manipulators are attached to grasp messy objects with the help of the 2 servo motors. Recognition of messy object is achieved by combining RFID algorithm, stereo vision and ultra-sonic.

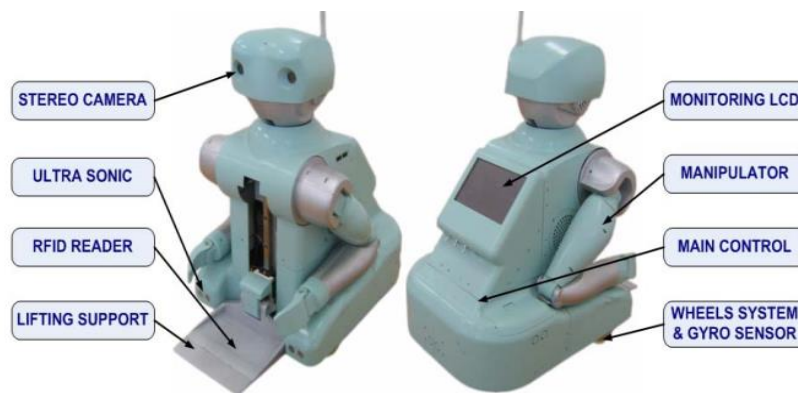


Figure 2-1 System configuration of McBot.

The McBot is made up of 3 main parts, agile autonomous navigation that enables efficient indoor navigation; novel manipulation system that allows McBot to manipulate target objects; intelligent sensing system helps McBot to recognise messy objects. By combining these systems, the robot can carry out tasks effectively. Figure 2.2 shows the McBot for task processing scenario.

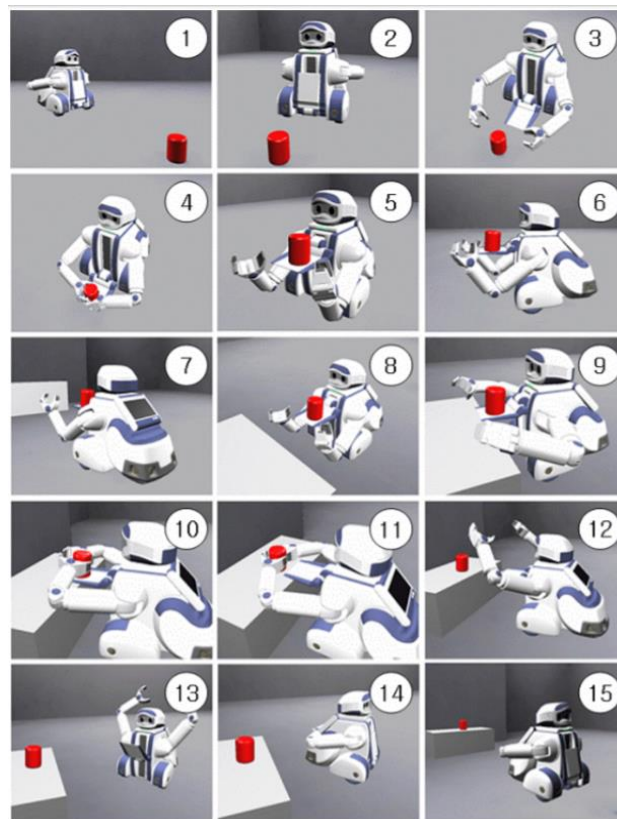


Figure 2-2 Task processing scenario of McBot.

1,2,3: Navigate and search for target object

4,5,6: Grasp target object

7,8,9: Move to destination spot

10,11,12: Put object at proper location

13,14,15: Search for next target object

McBot uses RFID navigation method which can avoid the line-of-sight problem that will obstruct navigation (Choi et al. 2006). A RFID reader is installed in McBot and the experimented indoor floor is tiled with multiple RFID tags associated with their respective coordinates. McBot will navigate around by detecting RFID tags that provides its coordinate to the navigation system.

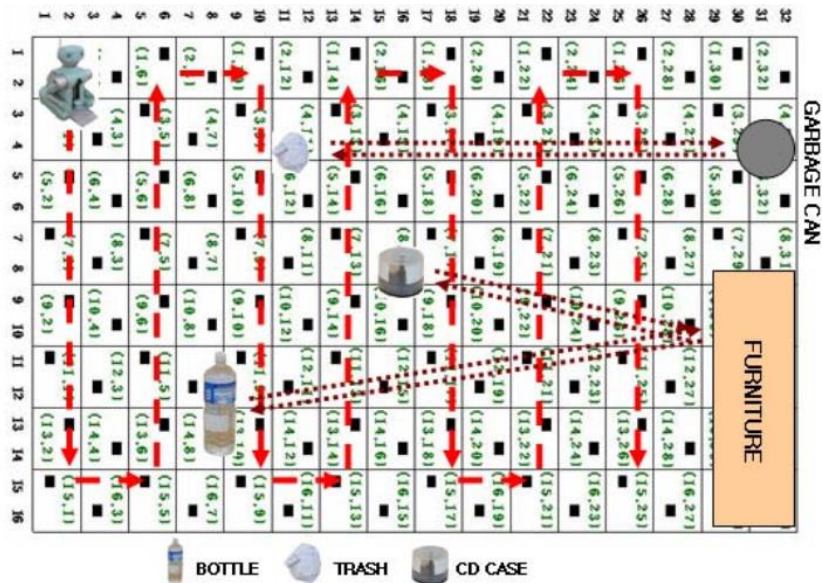


Figure 2-3 Floor equipped by RFID tags.

When deployed in a messy environment, the McBot can clear up the experimented indoor floor with decent efficiency. The average time taken for McBot to recognise and grab is about 18 seconds and the average time taken to arrange the item is about 20 seconds. However, McBot is heavily dependent on RFID tags to perform navigation and recognition of object, thus limiting its capability to interact with new objects and environments that are not RFID tagged. It also needs to tag the object to be arranged with RFID tag else it will be treated as trash and thrown away.

2.1.2 Autrobot

Other than that, the Autrobot developed by Kulkarni et al. (2013) is capable of autonomous indoor object detection and collection by applying indoor exploration algorithm with linear time complexity. Autrobot has 2 wheels connected a 3-element truss that forms a collecting arm. On the front end of Autrobot, 2 ultrasonic sensors are mounted. The indoor exploration algorithm aims to let Autrobot autonomously scan a confined area to search for trash objects and navigate towards the object for pick up by utilising ultrasonic sensors. Autrobot is not depending on a grid style area exploration mechanism (Sridharan 2008) or SLAM for mapping. Instead, it estimates object distance using two ultrasonic sensors. Autrobot will continuously monitor the ultrasonic sensor. When it detects the trash, it checks whether the object is stationary to add redundancy to the algorithm to identify trash object.

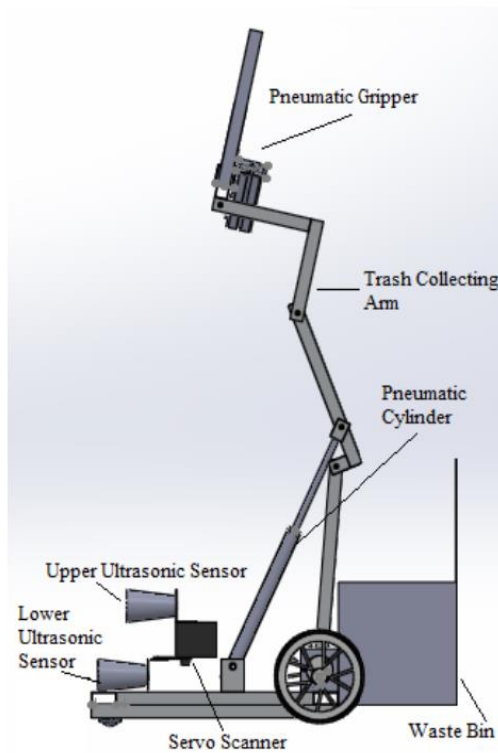


Figure 2-4 Autrobot

By using ultrasonic sensor for navigation, the navigation process will not be affected by colour and transparency of object. It is cheaper compared to other sensors and can operate in dark environments. On the other hand, the sensing accuracy will be degraded by soft materials such as indoor mat.

2.1.3 HERB



Figure 2-5 HERB

HERB is an autonomous mobile robot operating on a Segway that can perform manipulation tasks in household environment presented by Srinivasa et al. (2010). HERB is competent with humans when it comes to complicated manipulation tasks such as carrying a drink without spilling it. The robot is operating on the open source ROS platform for compatibility across multiple devices. Its sensing module hardware consist of multiple cameras and a planar laser on the Segway for motion planning, mapping and localisation. To track movable object in the environment, HERB uses GATMO, an approach that has been revised to improve dynamic mapping, object tracking, and object detection and recognition (Gallagher 2009).

For autonomous computer vision, HERB uses real-time object recognition and pose estimation from natural features such as object texture. The robot reconstructs as visual

model of objects in a virtual environment after executing its pose estimation algorithm. The pose information is later used in the grasping process. In order to plan for multiple task execution, for instance opening doors and throwing a soda can into a bin, manipulation planning is constructed alongside with vision system and multiple planning algorithm of the robot arm to succeed in tasks, specially tasks with constrains such as lifting a fully filled cup.

During demonstration of HERB, some of the failures faced during task execution can be linked to inconsistency of robot joints due to the use of cable-driven arm that is limited to ± 0.5 degrees, inaccuracy in camera-based robot localisation, and imprecision in Segway navigation that can prevent the robot arm from reaching the targeted object.

2.1.4 PR2

To improve the robot's justification in arrangement of objects, a solution is proposed by Abdo et al. (2016) is to let the service robot tidy up objects in containers through user preference prediction. They have shown that it is possible to calculate pairwise object preferences for recognised or unrecognised objects in the database based on previously or partially known preferences. Then they tried to construct a partitioning of object so that all pairwise constraints can be met. This approach allows robot to discover arrangement patterns from a sample of user preferences. The robot will be able to predict the preferred arrangement for objects by observing the previous arrangement made by the user. They conducted an experiment with the PR2 robot, where the robot is asked to arrange the 2 objects on table to shelves. The robot can place object with drinks object type together on a shelf, and object of food category together. The experiment result shows a success rate of 82% whereby the objects are placed orderly on empty shelves or grouped together with other objects with similarity.



Figure 2-6 Tidy up experiment with PR2.

The strengths of this approach are allowing robots to arrange any new objects by grouping objects of similar characteristics. The approach can satisfy most user preferences with minimal training sets of user preference. However, this approach requires many training sets in order to obtain accurate representation of user preference. Besides, if the user has a different preference compared to the training set, the robot will need more tries to get the correct result according to the user preference.

2.1.5 Johnny



Figure 2-7 Johnny

Johnny is an autonomous service robot for domestic environment developed by Breuer et al. (2012). The robot is created for cognitive tasks such as bringing a specific drink to a person in the house. To complete the task, Johnny needs to perform navigation, object detection and recognition, and object manipulation. For navigation, SLAM is utilised for autonomous navigation and self-localisation. ICE is used as the middleware to provide a platform for multiple operating system and programming language (Henning 2004). Other than that, Johnny is capable of people detection and tracking using data from the laser rangefinder for longer detection range.

Besides that, it is also able to differentiate 2 similar objects from one another by applying object categorisation first and then perform text mining to increase accuracy of object detection. For instance, a bottle ketchup and rat poison can appear to be similar in shape but there will be words like “tomato” or “delicious” appearing on the ketchup, but it is more unlikely to appear on the rat poison. Furthermore, Johnny can perform other tasks such as speech recognition, facial expression recognition and gesture recognition due to its numerous autonomous components implemented in Johnny.

2.2 Critical Remarks of Previous Works

Robot	Navigation method	Object Detection Sensor	Strength	Weakness
McBot	RFID sensing.	RFID, stereo camera, ultrasonic.	Wide coverage of task area without line-of-sight issue.	Need to manually layout RFID tags.
Autrebot	Ultrasonic sensing.	Object height satisfy defined height.	Simple implementation.	Requires manual object description.
HERB	Segway localization with GATMO.	Webcam.	Able to perform complex manipulation.	Manipulation planning algorithm is not yet perfect, errors can still occur.
PR2	Navigation planner modules.	RGB-D camera.	Object classification according to user preferences.	A solid prediction model is required
Johnny	SLAM with laser range finder.	Stereo camera.	Can perform multiple tasks.	Maximum payload is 500g.

Table 2-1 Comparison of existing service robot with tidy up functionality.

To fulfil the objectives of our project, this proposal presents a solution to develop a low-cost service robot with tidy up function using affordable hardware and open-source software. Our project is based on the TurtleBot3 Burger and ROS, which can provide a low cost, open source platform that is enough to complete the task requirement and has better scalability across different systems. Next, we are using an affordable RGB-D camera to achieve object detection and object tracking. However due to limited mounting space, instead of integrating the whole system in a single robot, we are going to separate the system into 2 parts. The first part is a robot with robot arm, wheels, camera. The camera is connected to the second part of the system, a laptop for image processing.

Chapter 3 System Design

3.1 Methodology

The general diagram of the proposed approach is shown in the figure below.

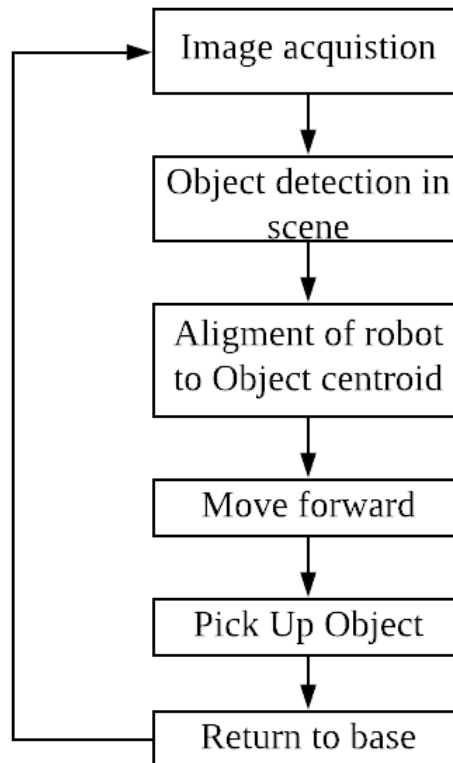


Figure 3-1 Methodology.

The system starts with image acquisition from the RGB-D camera, then object detection in scene is performed. When object is detected, the robot repositions itself to face the object's centroid. Then, the robot moves forward to the object's location slowly and perform pick up action. Finally, the robot brings the object back to base and repeating the search procedure.

3.2 Tools to Use

3.2.1 TurtleBot 3 Burger

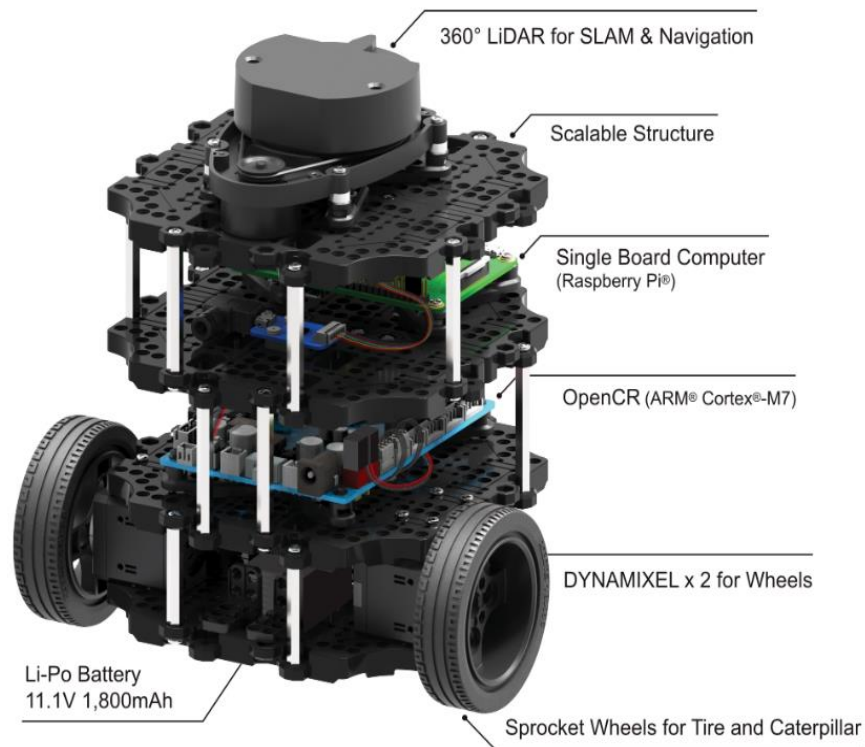


Figure 3-2 Components of TurtleBot3 Burger.

TurtleBot3 Burger is a small-sized, affordable, programmable, and expandable ROS-based mobile robot. The reason to use TurtleBot3 Burger for this project is to have a smaller platform and lower the budget without much performance price trade-off. TurtleBot3 Burger is equipped with 2 DYNAMIXELs in the wheel joints, to accurately obtain representation of spatial information for it to navigate around the workspace. Besides that, a manipulator can be installed to the TurtleBot to let it interact with objects in the workspace. It also comes with a single board computer (RPi B+ for our case), and an OpenCR board to control the robot arm for our project. The TurtleBot3 Burger can be remotely controlled from another PC via ROS nodes. Weighing at 1 kg, it can handle

payload weight of up to 15 kg. It can operate for about 2 hours and 30 minutes after a charging time of 2 hours and 30 minutes.

According to the ROBOTIS e-Manual (2018), TurtleBot3 is the most affordable robot compared to other mobile robots with similar technologies equipped. The TurtleBot3 Burger is highly customizable according to the requirements of the task by reconstructing its physical parts or installing additional components. TurtleBot3 Burger is compatible with existing ROS-based robot components, which makes it possible to extend its functionality. Besides that, with the open source hardware component, firmware and software framework of TurtleBot3, users are given the freedom to download, alter and share source codes. The included main controller board in TurtleBot, OpenCR, is an open source control module for ROS embedded systems. It can be configured through Arduino IDE to setup ROS publisher and subscriber nodes and control robot components connected to it such as the DYNAMIXELS and robot arm. On the other hand, TurtleBot can communicate with a remote machine by setting up ROS in TurtleBot's RPi. However, due to limitation on RPi's power capability, external hardware is being connected to the OpenCR.

3.2.2 Robot Arm

A 3-jointed robot arm with a gripper that consists of 4 servo motors from Futaba S3003 is being used as a manipulator for object manipulation in this project. This extension can compensate for the lack of manipulating ability in the original TurtleBot3 Burger model and can help complete its job as a tidy up service robot to interact with objects in the workspace.

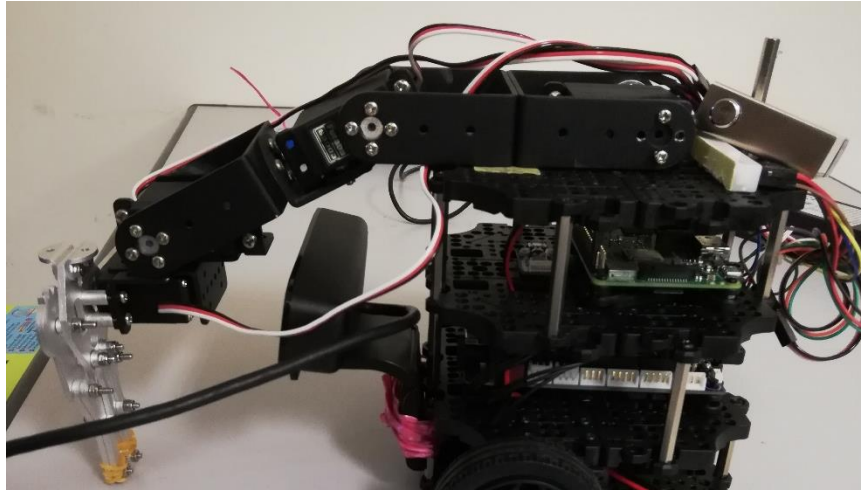


Figure 3-3 Robot Arm mounted on TurtleBot Burger.

3.2.3 RGB-D Camera

An RGB-D camera is required for object detection in this project. Hence, the sensor used for this purpose is the CREATIVE Senz3D depth and gesture recognition camera. By combining colour and depth details, better object detection results can be obtained. It also enables the robot to view the workspace and estimate the distance reachable by the robot arm. The RGB data output from the camera is 1280 by 720 pixels in size, whereas the depth data output has a resolution of 320 by 240 pixels with a depth resolution of 15cm to 1m, which is enough for close range object detection.



Figure 3-4 CREATIVE Senz3D.

3.2.4 ROS

ROS is an open source software framework for robot that provides communication between multiple machines that runs on ROS. For this project, ROS is used to establish communications between TurtleBot and remote PC that performs image processing and object detection functions. Besides that, ROS also provides low-level device control via data streaming through a topic. Therefore, we can control the hardware attached to the TurtleBot OpenCR through remote PC. Both TurtleBot and remote PC are communicating through ROS topic that contains information about publisher or subscriber nodes, and the data currently being published or subscribed in a topic.

3.2.5 OpenCV

OpenCV is an open source software library with various computer vision and machine learning algorithms. It is used extensively in research for many computer vision applications ranging from simple image processing to object detection in a scene. In this project, it is being used for input image pre-processing before applying its object detector on the input image from camera feed. The object detection model being used in this project is the LBP cascade algorithm that allows object detection by drawing bounding boxes around toy objects that are being detected in a video frame.

3.2.6 Arduino IDE

Arduino IDE is an open-source development environment for Arduino board control through Arduino programming. Since the project use the OpenCR board, which includes a connector that has same pin map as the Arduino Uno and is compatible with Arduino IDE for firmware development, it will be used to control robot arm and wheels through ROS nodes.

Chapter 4 Project Implementation

4.1 System Design

4.1.1 Hardware Block Diagram

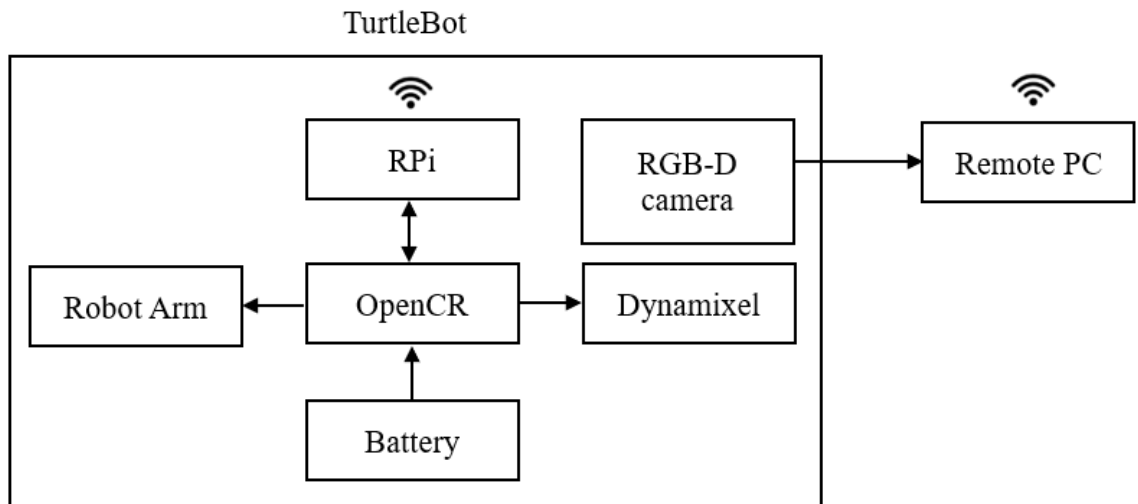


Figure 4-1 Hardware block diagram.

4.1.2 System Flowchart

The following diagram shows the stages involved in system execution phase, which are mainly image processing and robot control.

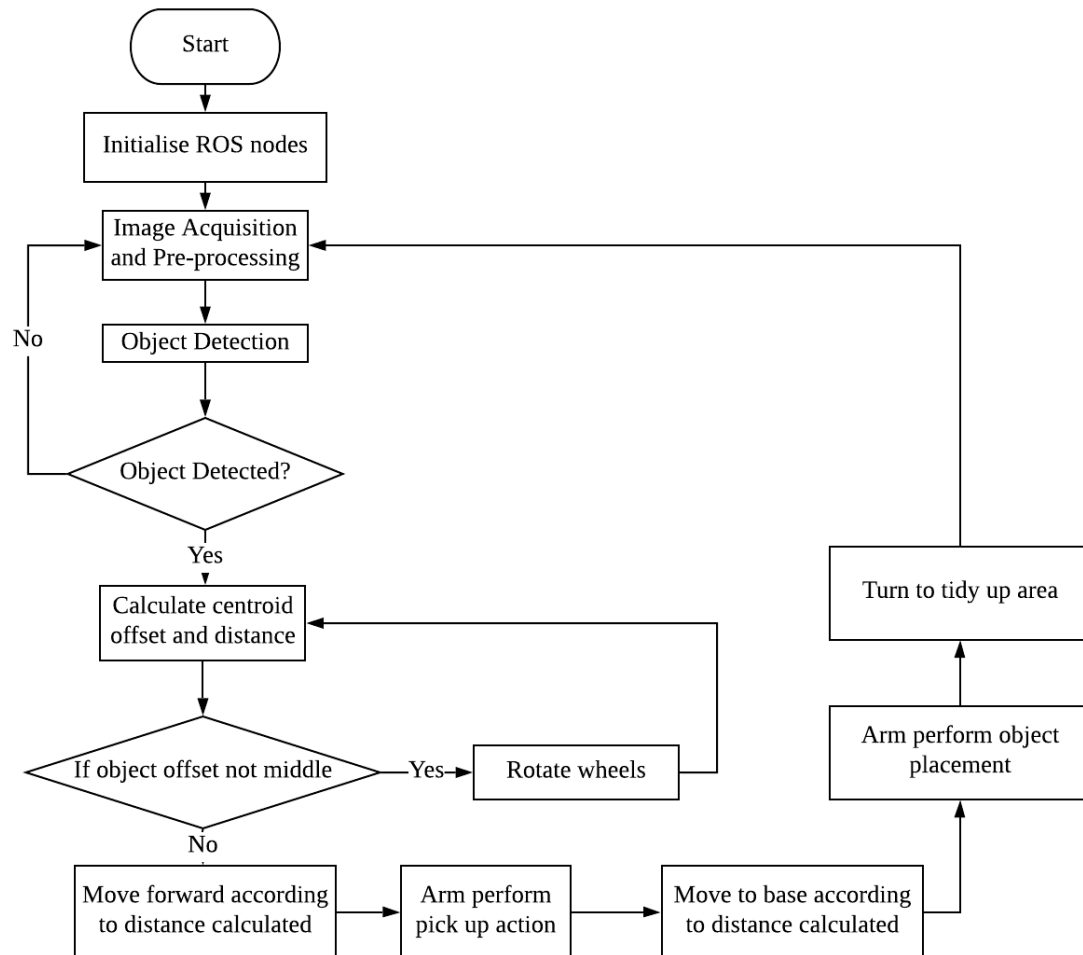


Figure 4-2 System Flowchart.

4.2 ROS Setup

This section will provide details for ROS setup in TurtleBot and remote PC.

4.2.1 ROS Setup on Remote PC

The remote PC will be using Ubuntu 16.04.6 LTS (Xenial Xerus) and ROS Kinetic Kame for better compatibility. We can either use a virtual machine or dual-boot the system to install Ubuntu. After that, we will proceed with installation of ROS and dependent ROS packages. The commands for the installation are shown in the figures below:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros_kinetic.sh
&& chmod 755 ./install_ros_kinetic.sh && bash ./install_ros_kinetic.sh
```

Figure 4-3 Install ROS on Remote PC.

```
$ sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard
ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino
ros-kinetic-rosserial-python ros-kinetic-rosserial-server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs
ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro
ros-kinetic-compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping
ros-kinetic-navigation ros-kinetic-interactive-markers
```

Figure 4-4 Install Dependent ROS Packages.

```
$ cd ~/catkin_ws/src/
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ cd ~/catkin_ws && catkin_make
```

Figure 4-5 Catkin make Dependent ROS Packages.

4.2.2 ROS Setup on TurtleBot3 Burger

An SD card with a storage size of 16GB is used as a bootable media with Raspbian image file to install Linux on TurtleBot RPi. After setting up successfully, we will be able to boot into TurtleBot's RPi. The software for ROS and TurtleBot3 is pre-installed in the Raspbian. To bring up the TurtleBot, simply SSH to the RPi's IP address and start the command as shown in the picture below.

```
liewzy@liewzy-Lenovo-Z40-70:~/Desktop$ ssh pi@192.168.43.222
pi@192.168.43.222's password:
Permission denied, please try again.
pi@192.168.43.222's password:
Linux raspberrypi 4.19.42-v7+ #1219 SMP Tue May 14 21:20:58 BST 2019 armv7l
```

Figure 4-6 SSH to RPi.

4.2.3 Network Configuration

To connect both devices via Internet, we need to setup both devices by editing the `.bashrc` file. The current IP address of the device can be shown with the “`ifconfig`” command in Terminal window. To access the `.bashrc` file, the command “`nano ~/.bashrc`” is entered in Terminal window:

For TurtleBot, setup of ROS connection is done by inserting the following lines to the `.bashrc` file:

```
export ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
export ROS_HOSTNAME = IP_OF_TURTLEBOT
```

For the remote PC, the following lines are inserted in `.bashrc` file:

```
export ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
export ROS_HOSTNAME = IP_OF_REMOTE_PC
```

After inserting the lines, the .bashrc file is sourced by the command “`source ~/.bashrc`”.

4.3 Hardware Interfacing and Setup

The figure shows the hardware interface of the TurtleBot from front view and side view.

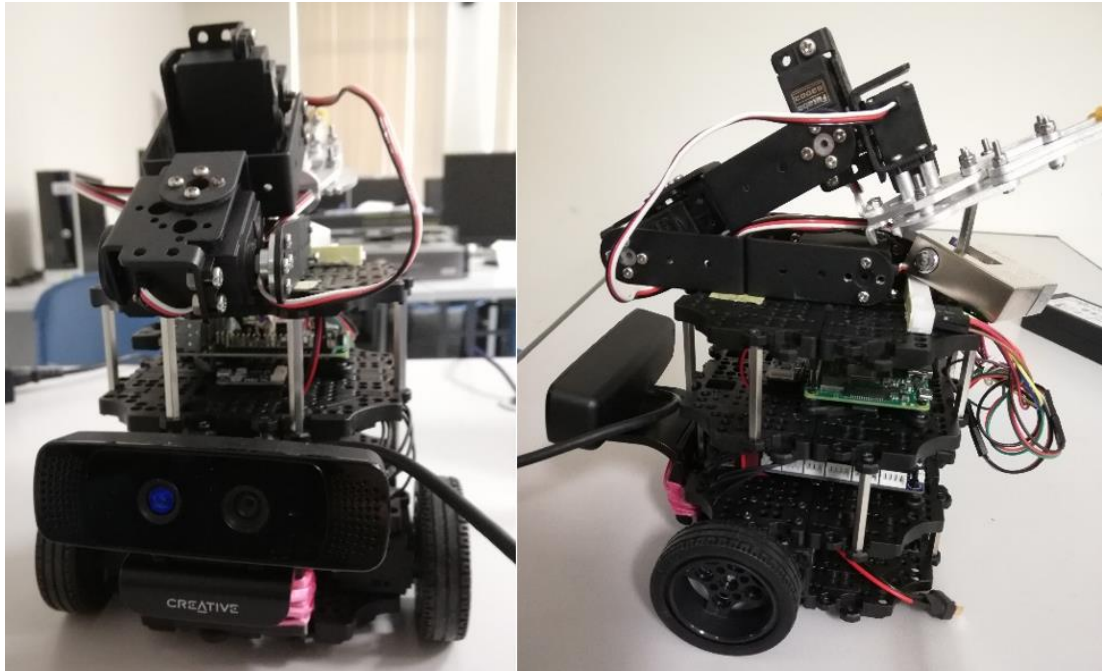


Figure 4-7 Implemented System Front View (left) and Side View (right).

4.3.1 OpenCR Setup

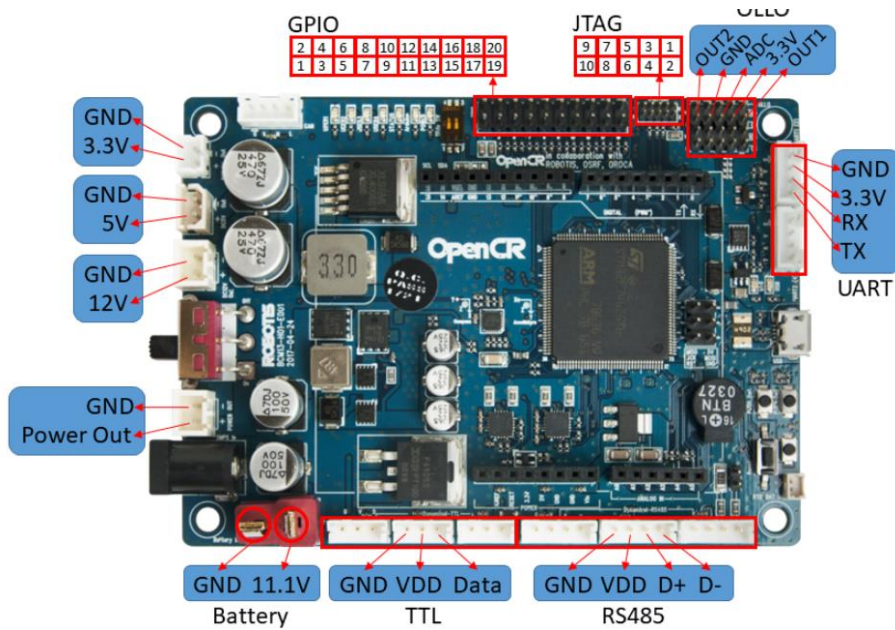


Figure 4-8 OpenCR Pin Map.

The robot wheels are connected to the OpenCR board's TTL pins whereas the robot arm will be connected to the Arduino digital output pins, 5V pins and ground pins. From previous figure 4-7, the servo motors are labelled from 1 to 4 starting from the lowest servo motor to the servo motor at the gripper. To setup publisher and subscriber nodes that publish data from remote PC to robot arm and robot wheels, the firmware of OpenCR is modified. To modify the firmware and upload it to OpenCR, Arduino IDE will be installed on remote PC with the following commands. This first section sets up remote PC's USB port for access without root permission and the last line sets up 32-bit compiler for OpenCR libraries:

```
$ sudo cp ./99-opencr-cdc.rules /etc/udev/rules.d/  
$ sudo udevadm control --reload-rules  
$ sudo udevadm trigger  
$ sudo apt-get install libncurses5-dev:i386
```

Figure 4-9 Allow USB port access and 32-bit compiler.

After those steps, we can download and install Arduino for Linux system from the Arduino website. We will be able to access turtlebot3_core firmware from File -> Examples -> turtlebot3 -> turtlebot3_burger tab. This section shows modification made to the OpenCR firmware.

```
void setup()  
{  
  DEBUG_SERIAL.begin(57600);  
  
  // Initialize ROS node handle, advertise and subscribe the topics  
  nh.initNode();  
  nh.getHardware()->setBaud(115200);  
  
  nh.subscribe(arm_control);  
  nh.subscribe(cmd_vel_sub);  
}
```

Figure 4-10 Adding subscriber node in turtlebot3_core.ino


```
void armControlCallback(const std_msgs::Byte& toggle_msg)
{
    //0: initial arm position
    if (toggle_msg.data == 0){
        servo2.write(150);
        servo3.write(110);
        servo4.write(180);
    }

    //1 pick up
    else if(toggle_msg.data == 1){
        servo4.write(100); //open gripper
        delay(100);
        servo2.write(5); //adjust for height
        servo3.write(45); //adjust for distance
        delay(1000);
        servo4.write(180);
        delay(100);
        servo3.write(50);
        delay(500);
        servo2.write(150);
    }

    //2
    else if (toggle_msg.data == 2){
        //lower down upper arm
        servo2.write(25); //adjust for height
        servo3.write(40); //adjust for distance
        delay(1000);
        servo4.write(100);
        delay(1000);
        servo2.write(150);
        servo3.write(110);
        servo4.write(180);
    }
}
```

Figure 4-11 Robot arm control function in turtlebot3_core.ino

```

#include <math.h>
#include <Servo.h>

void initServo(void){
  servo2.attach(5); //0(front)- 180
  servo3.attach(6); //0(front)- 180(back)
  servo4.attach(9); //100(open) - 180(close)

  //initial arm position
  servo2.write(150);
  servo3.write(110);
  servo4.write(180);
}

// Callback function prototypes
void commandVelocityCallback(const geometry_msgs::Twist& cmd_vel_msg);
void soundCallback(const turtlebot3_msgs::Sound& sound_msg);
void motorPowerCallback(const std_msgs::Bool& power_msg);
void resetCallback(const std_msgs::Empty& reset_msg);
void armControlCallback(const std_msgs::Byte& toggle_msg);

```

Figure 4-12 Function callback in turtlebot3_core_config.h

To upload the code to OpenCR, the OpenCR board is connected to remote PC via micro USB cable and by clicking the upload button as shown in figure 4-13. If the upload is successful, the “**jump_to_fw**” message will be displayed in the terminal as shown in figure 4-14.

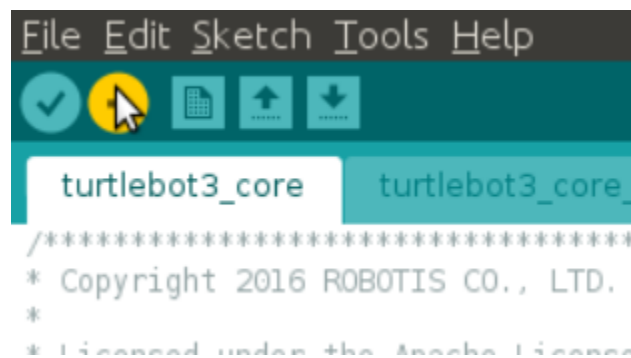
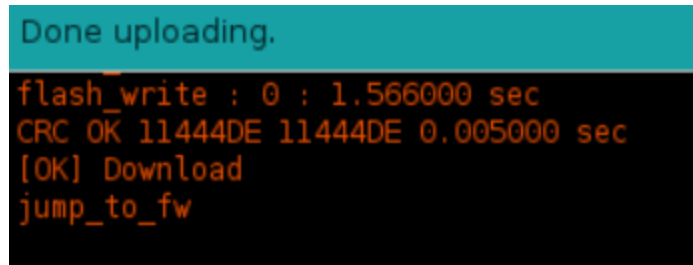


Figure 4-13 Upload firmware to OpenCR.

A terminal window showing the output of a firmware upload process. The text is as follows:

```
Done uploading.  
flash_write : 0 : 1.566000 sec  
CRC OK 11444DE 11444DE 0.005000 sec  
[OK] Download  
jump_to_fw
```

Figure 4-14 Successful firmware upload to OpenCR.

4.3.2 Bring Up TurtleBot

To basic packages to start TurtleBot applications to setup publisher and subscriber nodes in TurtleBot, we will be using the command listed below.

On remote PC: `roscore`

On TurtleBot: `roslaunch turtlebot3_bringup turtlebot3_robot.launch`

If the bring up process is successful, the terminal on TurtleBot will show the message “Calibration End”.

4.3.3 RGB-D Camera Setup

DepthSense SDK and Softkinetic camera driver are the necessary tools in order to use the camera’s depth sensor. Since the Softkinetic camera driver is only available on Linux platform, we connected the camera to remote laptop instead of RPi on TurtleBot. The DepthSense SDK can be downloaded from the website:

https://github.com/robotican/komodo_2.0/blob/master/komodo2/third_party_files/DepthSenseSDK-1.9.0-5-amd64-deb.run

After downloading the file, the DepthSense SDK is installed to the remote PC by typing these commands to the terminal:

```
chmod u+x DepthSenseSDK-1.9.0-5-amd64-deb.run
sudo ./DepthSenseSDK-1.9.0-5-amd64-deb.run # accept agreement
```

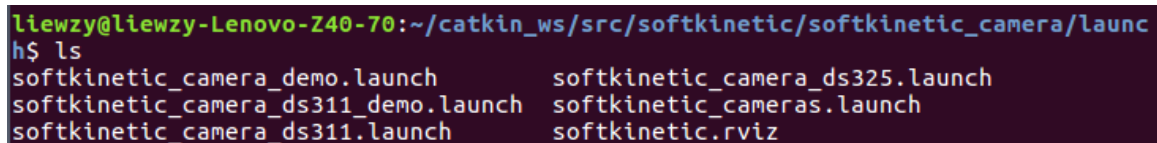
The agreement is accepted by pressing ‘q’, then the path “/opt/softkinetic/DepthSenseSDK” will be accessible to user. The path to the DepthSense SDK can be exported to .bashrc by entering the command:

```
export LD_LIBRARY_PATH=/opt/softkinetic/DepthSenseSDK/lib
```

Next, to install Softkinetic driver, first we retrieved the driver file from:

```
https://github.com/ipa320/softkinetic
```

Then the file is extracted and moved to “~/catkin_ws/src/” directory and “catkin_make” is executed to build and install the package. Later, we could see the RGB-D image data being published to ROS topic by running “roscore” first and “roslaunch softkinetic_camera softkinetic_camera_ds325.launch”. The topic published will be named the same as the package name that we just built with “catkin_make”. The topic published could be observed by typing “rostopic list” in the terminal.



```
liewzy@liewzy-Lenovo-Z40-70:~/catkin_ws/src/softkinetic/softkinetic_camera/launc
h$ ls
softkinetic_camera_demo.launch      softkinetic_camera_ds325.launch
softkinetic_camera_ds311_demo.launch softkinetic_cameras.launch
softkinetic_camera_ds311.launch     softkinetic.rviz
```

Figure 4-15 Softkinetic camera launch files location.

4.4 Custom Object Detector

A custom object detector is required for this project to detect toy objects in a scene. To train a custom object detector, we have collected around 100 image datasets of toy objects (plastic dinosaurs) as positive samples and around 2000 images in computer laboratory as negative samples.



Figure 4-16 One of the positive samples.

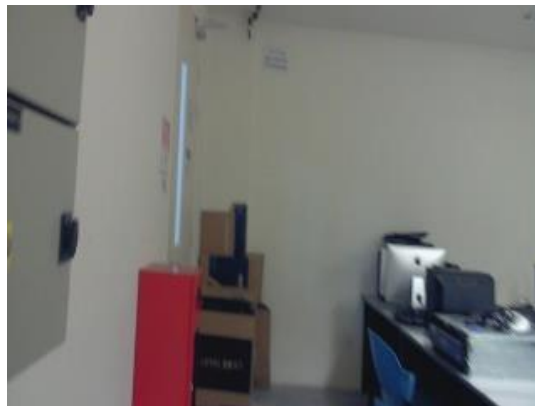


Figure 4-17 One of the negative samples.

The next step is to create descriptor file for the negative samples and specify the number of positive images (pos) and negative (neg) images to be used in training for a specified number of training stages (num). This requires a lot of trial and error, but the rule of thumb is to use a double the amount of positive image samples for negative image samples.

```
def create_bg(neg_img_dir):
    for img in os.listdir(neg_img_dir):
        line = neg_img_dir + "/" + img + "\n"
        with open("bg.txt", "a") as f:
            f.write(line)
    f.close()
```

Figure 4-18 Generate text file to index all negative samples.

```
def merge_info_lst():
    for infolst in os.listdir('positives'):
        if "info" in str(infolst):
            with open("positives/"+infolst) as f:
                with open("info.lst", "a") as fl:
                    for line in f:
                        fl.write(line)
            os.remove("positives/"+infolst)
    fl.close()
    f.close()
    os.system("mv info.lst positives/")
```

Figure 4-19 Generate info.lst by merging images.

```
opencv_traincascade -data output_cascade
-vec positives.vec
-bg bg.txt
-numPos pos
-numNeg neg
-numStages num
-w 25 -h 25
-featureType LBP
```

Figure 4-20 Start LBP cascade training.

4.5 Main Program

The following subsections will provide details on main program functions.

4.5.1 Image Topic Subscription

This section will setup the subscriber node to receive video data from the RGB-D camera we setup earlier. The image is in ROS sensor_msgs format, we will use cv_bridge to convert it to OpenCV's Mat format.

```

class Detector:
    def __init__(self):
        rospy.init_node('TB3_detect_obj')
        rospy.loginfo("Turtlebot Detection Starting...")
        #publisher setup
        self.velocity_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
        self.arm_pub = rospy.Publisher('arm', Byte, queue_size=10)
        #subscriber setup
        self.color_sub = message_filters.Subscriber('softkinetic_camera/rgb/image_color', Image)
        self.depth_sub = message_filters.Subscriber('softkinetic_camera/depth/image_raw', Image)
        ts = message_filters.ApproximateTimeSynchronizer([self.color_sub, self.depth_sub], 1, 0.5)
        ts.registerCallback(self.RGBD_callback)
        self.bridge = CvBridge()
        self.distance = 0
        self.pickedUp = False
        self.vel_msg = Twist()

    def RGBD_callback(self, rgb, d):
        #cv_bridge conversion to Mat and resize to match depth image
        self.color_image = self.bridge.imgmsg_to_cv2(rgb, "bgr8")
        self.depth_image = self.bridge.imgmsg_to_cv2(d, "32FC1")

        self.process()

```

Figure 4-21 Subscribing to topics published by Softkinetic camera.

4.5.2 Image Pre-processing

The code snippet for image pre-processing is shown in figure 4-18, which produces the result in figure 4-19 by filtering out image background that are not in vibrant colors like toys by trial and error method with hue, saturation, and value in the image.

```

color_resize = cv.resize(self.color_image, None, fx=0.5, fy=0.5)
#image pre-processing
hsv = cv.cvtColor(color_resize, cv.COLOR_BGR2HSV) #cv loads image in bgr
l_hsv = np.array([0, 70, 0])
u_hsv = np.array([200, 255, 255])
hsv_mask = cv.inRange(hsv, l_hsv, u_hsv)
result = cv.bitwise_and(color_resize, color_resize, mask= hsv_mask)

```

Figure 4-22 Performing image pre-processing.



Figure 4-23 Resulting resized image after colour masking.

4.5.3 Object Detection in Input Image

The object detector is set to return the first detected toy object and then proceeds to calculate object centroid offset from the camera centre. Once the object detector has detected the toy object, the centroid offset of the bounding box around the detected object is calculated and the coordinates of the area around the centroid is mapped to the depth image to calculate the average distance of the object. Then, the robot control part will be started.


```

#run detection model on segmented image
toys = cascade.detectMultiScale(result, 1.3, 5)

if(len(toys) != 0):
    #limit to first item to lower false detection
    x = legos[0][0]
    y = legos[0][1]
    w = legos[0][2]
    h = legos[0][3]

    cv.rectangle(color_resize, (x,y), (x+w,y+h), (255,0,0), 2)
    object_mid = ((x+x+w)*0.5, (y+y+h)*0.5)

    #get depth of center area
    object_mid_depth = (int((object_mid[0]-30)/2), int(object_mid[1]/3*2)+30) #x, y
    roi_depth = self.depth_image[object_mid_depth[1]-10: object_mid_depth[1]+10,
                                object_mid_depth[0]-10:object_mid_depth[0]+10]

    #x, y
    count = 0
    avg = 0.0

    #discard nan data
    for x in range(len(roi_depth[0])):
        for y in range(len(roi_depth[1])):
            if ~np.isnan(roi_depth[x][y]):
                avg += roi_depth[x][y]
                count += 1
            else:
                continue

    if count > 0:
        avg = avg/count
        offset = object_mid_depth[0] - center_d[0]
        rospy.loginfo('Object detected at offset: %s, distance: %scm', offset, avg)

```

Figure 4-24 Object detector.

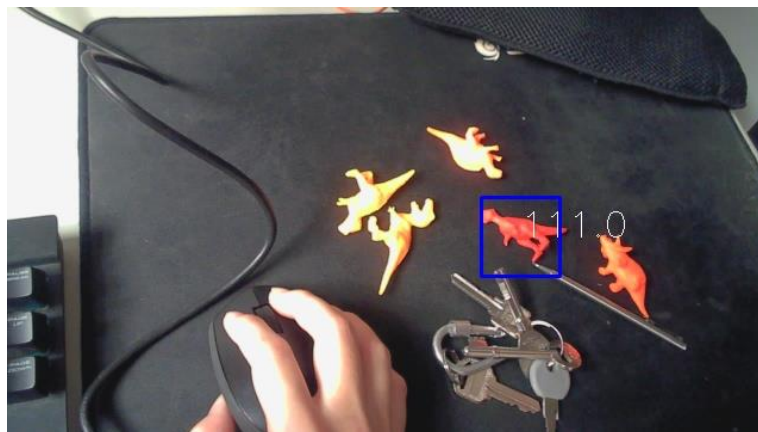


Figure 4-25 Object detection returning first detected result.

4.5.4 Turning Robot to align midpoint to object centroid

The robot will reposition itself to align to object centroid. Once that is done, it will retrieve the object's depth data from the depth sensor.

```
if offset < -10:  
    self.vel_msg.angular.z = 0.1  
    #rospy.loginfo('Going left')  
    self.velocity_pub.publish(self.vel_msg)  
    rospy.sleep(0.1)  
else:  
    if offset > 10:  
        self.vel_msg.angular.z = -0.1  
        #rospy.loginfo('Going right')  
        self.velocity_pub.publish(self.vel_msg)  
        rospy.sleep(0.1)  
    else:  
        #done centering,go forward now  
        self.vel_msg.angular.z = 0  
        self.velocity_pub.publish(self.vel_msg)  
        rospy.sleep(0.1)  
        rospy.loginfo('Stopping rotation, starting to move towards object at %s away', avg)
```

Figure 4-26 Turning TurtleBot sideways.

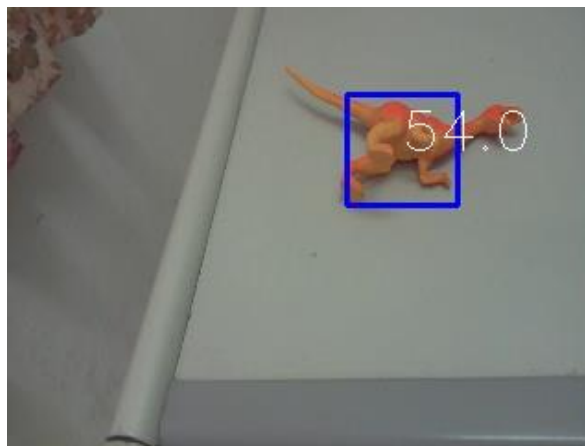


Figure 4-27 Object detected has slight offset to right.

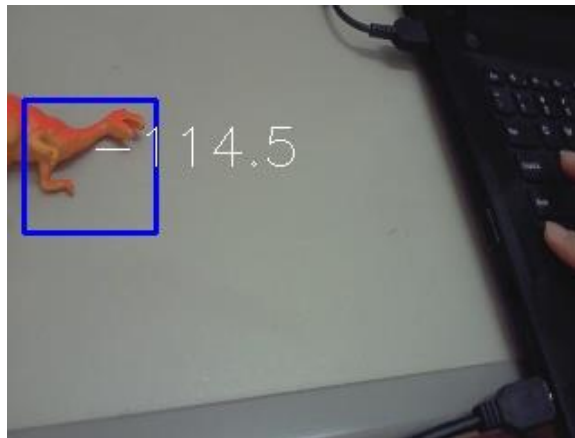


Figure 4-28 Object detected has slight offset to left.



Figure 4-29 Object detected at middle and its depth profile.

```
Terminal File Edit View Search Terminal Tabs Help
liewzy@liewzy-Lenovo-Z4... x liewzy@liewzy-Lenovo-Z4... x lie
0.235165
obj at middle, detecting distance to navigate
0.235515
obj at middle, detecting distance to navigate
0.234939
obj at middle, detecting distance to navigate
0.234086
obj at middle, detecting distance to navigate
0.231881
obj at middle, detecting distance to navigate
0.230771
obj at middle, detecting distance to navigate
0.229285
obj at middle, detecting distance to navigate
0.230263
obj at middle, detecting distance to navigate
0.23025
obj at middle, detecting distance to navigate
0.230668
```

Figure 4-30 Depth data output.

4.5.5 Pick Up Object at Object Location

```

current = 0.0
self.vel_msg.linear.x = 0.1

t0 = rospy.Time.now().to_sec()

self.distance = round(avg, 2) - 0.16 #deduct distance for arm pickup (10cm)

while(current < self.distance):
    rospy.loginfo("Going forward %sm, current distance: %s", self.distance, round(current, 2))
    self.velocity_pub.publish(self.vel_msg)

    t1=rospy.Time.now().to_sec()
    current = round(self.vel_msg.linear.x*(t1-t0),2)

#stop after reached
rospy.sleep(0.1)
self.vel_msg.linear.x = 0
self.velocity_pub.publish(self.vel_msg)
rospy.sleep(1)

rospy.loginfo("Picking up object...")
self.arm_pub.publish(1)
self.pickedUp = True

```

Figure 4-31 Going forward to pick up object.

4.5.6 Returning to Base

After picking up the object, the robot will go backwards to return the object to the base, which is the initial starting point of the robot.

```

rospy.loginfo('Prepare to go to goal now')
self.vel_msg.angular.z = 1
self.velocity_pub.publish(self.vel_msg)
rospy.sleep(4.6)
self.vel_msg.angular.z = 0
self.velocity_pub.publish(self.vel_msg)
rospy.sleep(2)

```

Figure 4-32 Returning to base.

4.5.7 Placing Object at Base and Returning to Search for Toy Object

Upon reaching the base, the robot arm will place the object and the robot will turn around to continue searching for new toy objects.

```
#drop and reset status
self.arm_pub.publish(2)
rospy.sleep(5)

rospy.loginfo("I have dropped the object, turning back")
self.vel_msg.angular.z = -1
self.velocity_pub.publish(self.vel_msg)
rospy.sleep(3)
self.vel_msg.angular.z = 0
self.velocity_pub.publish(self.vel_msg)
rospy.sleep(3)
```

Figure 4-33 Placing object and continue search.

4.6 Visualisation of ROS Computational Graph

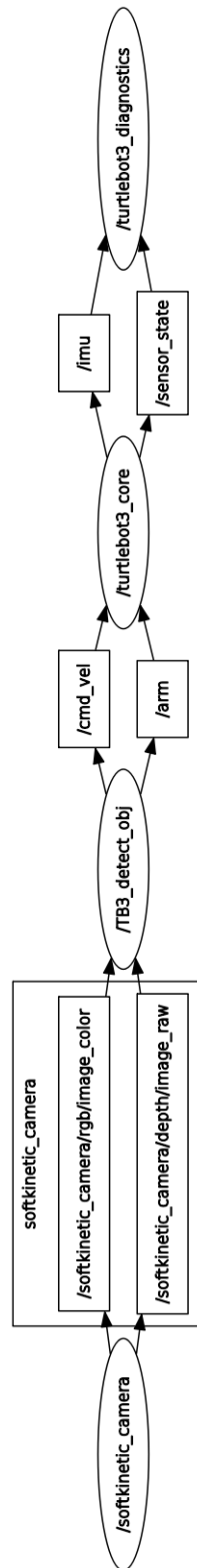


Figure 4-34 rqt_graph of system.

4.7 Implementation Issues and Challenges

Some difficulties arise during the project development, one of it is the integration of each individual hardware into a single system. Unlike the other home service robot mentioned in literature review, which has successfully embedded all individual parts into a robot, the TurtleBot3 Burger has limited space to mount all of the equipment, therefore we need to separate the robot into 2 parts: TurtleBot 3 Burger with the robot arm and RGB-D camera for object detection; and a remote PC connected to the camera via USB cable for camera input image processing. However, the camera is connected to the remote PC instead of RPi due to driver compatibility issue. During implementation of object recognition task, challenges occur when it comes to defining the proper parameters to filter out region of interest for object recognition target. The image should contain enough information for object classification while keeping the noise level low. Besides, object detection using machine learning has not reached its optimal performance due to limitation on datasets and tuning of detection parameters. It is also the nature of LBP cascades to give more false positive results as compared to Haar cascades even though it has a lower detection time.

As for robot control, manipulation of toy objects is limited by the gripper's opening distance, which is up to 5cm, hence it is likely for the robot to fail to pick up the object when the body of the object is too wide or the object is too small as it will slip away from the tiny gaps between the gripper. Furthermore, since the robot arm is about the same weight as the TurtleBot, weight balancing is required to prevent the robot from toppling when picking up items. Therefore, we added a weight at the back of the robot. We also need to prevent the robot from running over small objects that will obstruct the robot's movement. Therefore, a fixed amount of distance between toy object and the robot is added to the object distance calculation to resolve the issue. Besides that, a robust algorithm is required for machine learning in this project for better result. Another challenge faced in this project would be the way to integrate the data from sensor feedback coming from the robot with the processed data from remote PC. In order to control the TurtleBot3 from the remote PC, thus we must properly setup a platform to receive image data sending from the

Chapter 4 Project Implementation

TurtleBot3 and then feedback the required actions to the corresponding robot components (robot arm and wheels) after processing.

Chapter 5 System Testing

5.1 Test Scenario: Tidy Up Area with Single Toy Object

This scenario demonstrates the execution of tidy up task on a toy object (orange dinosaur toy) being placed in front of the robot, which is the area to perform tidy up. The toy object is not aligned to the robot arm pick up point as the camera was facing towards the right and it is out of robot arm's pick-up range. This test scenario is testing for the robot's ability to detect the object within its field of view and adjust the robot's position accordingly to align the toy object to the camera's centre point for it to be picked up later. Then, the robot is tested for its motion to move forward towards the object until the toy object has reached the arm pick-up point, which will be about 15cm away from the wheels. After the robot reached the toy object, the robot arm should pick up the object, the robot will then turn 180 degrees backwards and travel the same distance back to its initial point to drop off the item. The process is repeated until there is no to objects left in the area.

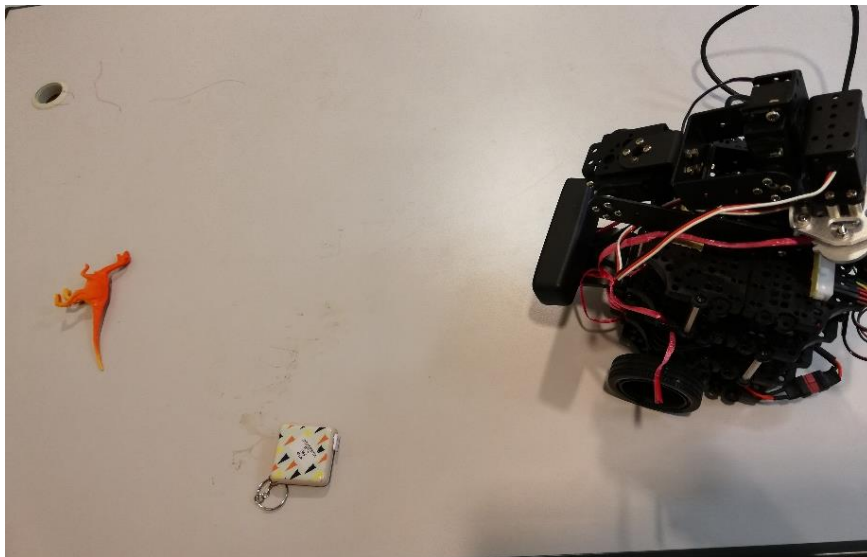


Figure 5-1 Initial point for tidy up task.

5.2 Test Result

From figure 5-2, we can see that the robot could reposition itself to center the toy object. However, the process can take a long time due to lower video frame rate from active object detection, hence the position of toy object in the received frame might not be the latest position in real time. Despite that, the robot eventually settles down and the robot arm is aligned to the toy object.

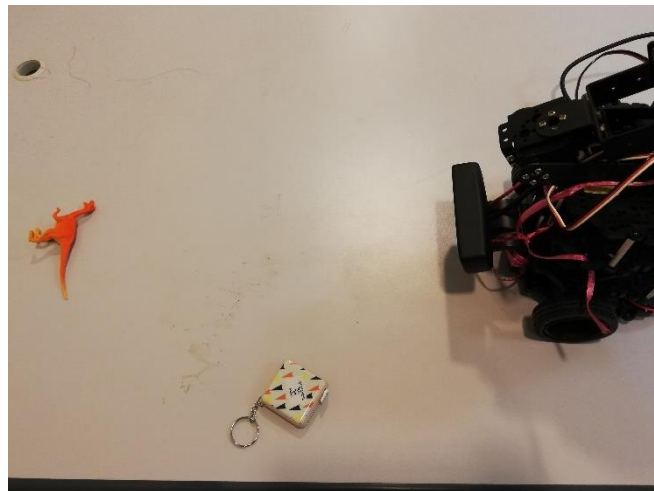


Figure 5-2 Robot repositioned to face toy object.

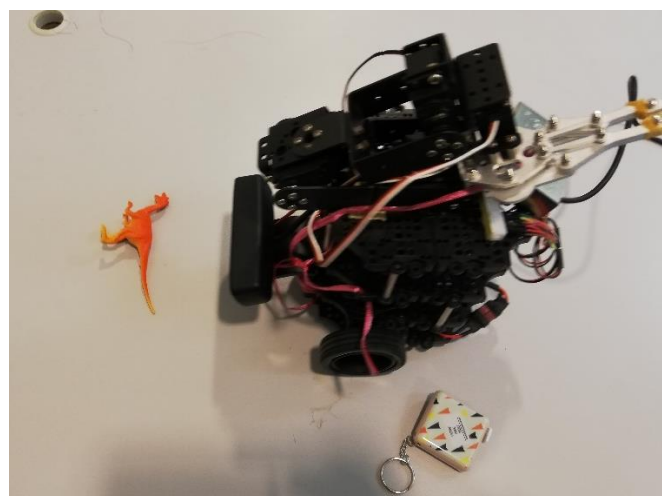


Figure 5-3 Robot moving forward until toy object enters arm pick-up range.

Once the robot detected that the toy object is aligned to the centre, it started to move forward until the object is in the arm pick-up point, as shown in figure 5-3. Then, it prepares to grab the toy object. However, due to the way the object was positioned, the arm could not fully grab the toy object. We had to manually rotate the toy object as shown in figure 5-4 to allow the robot arm to grab the object. Even with user help, the bounding box predicted is not the main body of object, making it hard to grab. In figure 5-6, the robot arm could pick up the object and try to return the object to its starting point.

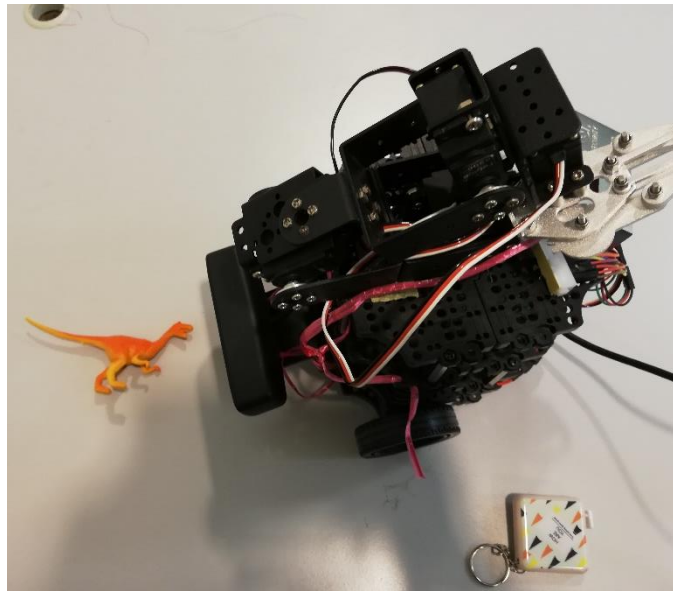


Figure 5-4 Rotated toy object due to limited gripper opening.

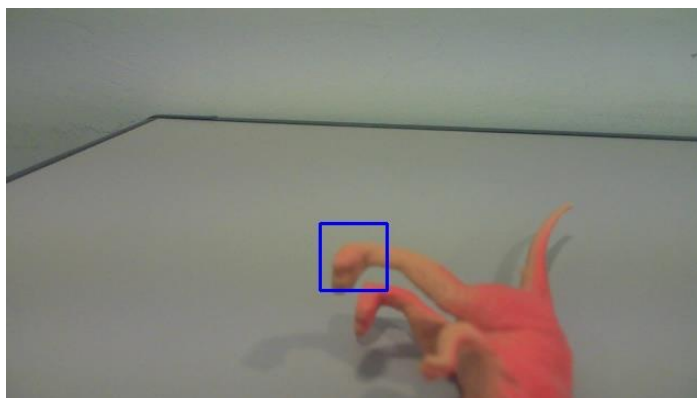


Figure 5-5 Small bounding box issue.

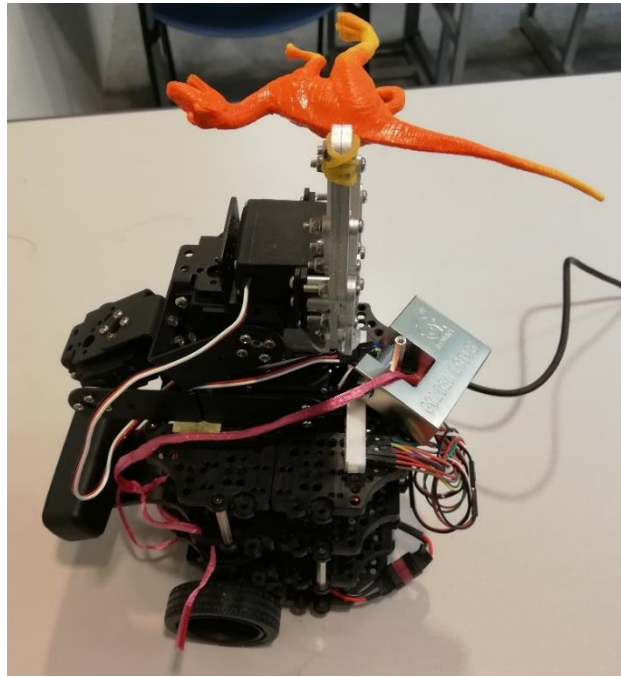


Figure 5-6 Successful attempt on picking up toy.

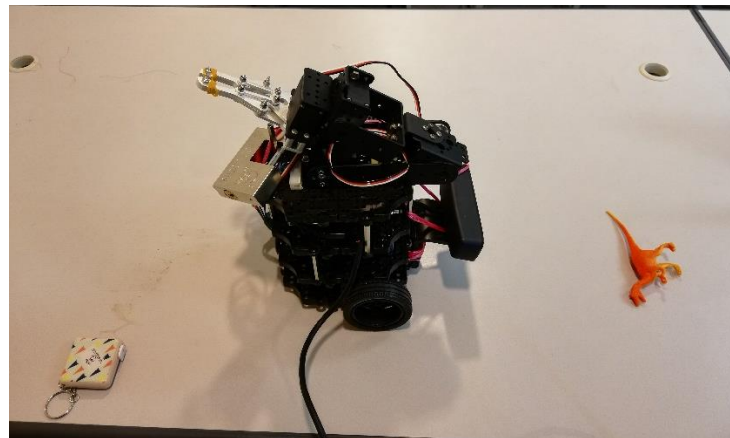


Figure 5-7 Placing toy objects to initial point.

Finally, the robot had successfully returned the object to its initial point, dropped the object, return to the same position as in figure 5-1 to search for other toy objects in tidy up area. Yet, false detection occurs from time to time and the robot will be returning the wrong item to the collection area. Figure 5-8 shows an example of false positive detection when the robot performs object detection in a scene.

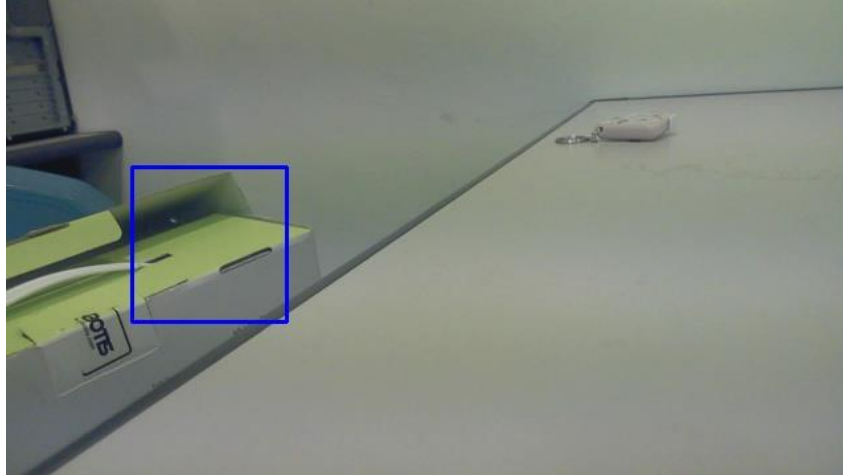


Figure 5-8 False positive detection.

5.3 Result Discussion

Overall, the result shows the TurtleBot can perform the task to pick up the toy item and return it to the box. However, the results for toy detection do not show high accuracy since false detection can occur with non-toy objects that have similar shapes as the toy objects. This is a result of using LBP cascade object detection algorithm, which is susceptible to lighting conditions and false detection. The ways to improve this customised object detector is to train with more datasets, preferably 1000 images for each category of toys, and improvising the object detector through feedback of falsely detection scenes to the negative datasets. Moreover, continuous detection from video frames will introduce flickering effect on bounding boxes and inconsistent bounding boxes sizes. To reduce erroneous result, object detector only returns the first identified toy objects from the scene.

Besides, the TurtleBot arm cannot adjust its gripper position for better grasp on the toy object when the toy object is further or nearer to the arm pick up point. Due to the inconsistent depth data from depth sensor, which has a minimum depth detection range of 10cm when attached to the robot's bottom layer, we are not able to retrieve precise object distance and adjust the robot arm's degree of movement accordingly. In addition, the camera's maximum depth range is up to 1m, so distance measurement for toy objects that lies outside the detection range is not possible. The robot could move forward until the

object enters the detection range, but the object detector cannot detect far objects accurately since the camera resolution is limited to 1280 * 720 pixels. Due to lack of robot navigation implementation, the movement of robot to the goal is based on experimenting with various velocity and publish time, so it might result in inaccurate robot movement such as overshooting when trying to stop in front of the object or trying to turn 180 degrees backwards after robot arm has performed its action. Lastly, the robot is not capable of navigating around obstacle to reach the tidy up area due to lack of navigation methods and reliance on camera input for movement.

Chapter 6 Conclusion

6.1 Project Review

In conclusion, the project aims to solve the current problem of household organisation which requires a lot of resources for regular maintenance. This project will provide a better solution for autonomous household organisation compared to the currently popular vacuum cleaner robot that only carries out simple functions such as swiping, mopping and vacuuming. The product developed is an affordable indoor service robot that can perform tidy up function to clear up a messy location with toy objects autonomously. The robot is installed with a robot arm and an RGB-D camera for object detection and manipulation. By utilising the RGB-D camera data, object tracking can be carried out on detected toy objects for simple robot motion planning. Besides that, toy objects can be categorised and placed back to the respective box. Overall, this project provides a low-cost solution for users who wish to have an autonomous service robot to tidy up their household.

6.2 Future Work

The project could be improved from object detection and robot navigation aspects. For object detection, further improvements can be made on the custom object detector with LBP cascade by providing larger datasets, through larger collection of toy objects image or performing image data augmentation and reuse false detections as negative datasets. Alternatively, deep learning object detection models such as TensorFlow or YOLO could be used as a better, more accurate, and responsive object detector model, provided the machine for processing is relatively powerful. Since the camera used in this project has limited depth detection range and the camera driver is not compatible with ARM Architecture in TurtleBot's RPi, improvements could be done by changing the camera for a compatible camera or expanding the robot with extra mounting points for the machine performing image processing. Then, a remote service robot can be truly simulated without navigation limitations from externally connected cables.

Other than that, robot control could be improved by implementing extra detection algorithm to determine whether the object of that size could be picked up by the arm, and if the object's body cannot be fitted in the gripper, the algorithm would detect the points that can be grabbed by the gripper. Additionally, robot navigation could be implemented in the future to provide accurate navigation methods. SLAM is state-of-the-art robot navigation method for TurtleBot. With a navigation method, the robot would be able to map the surrounding, perform navigation and localisation accurately autonomously.

Bibliography

Abdo, N, Spinello, L, Burgard, W & Stachniss, C 2016, 'Organizing objects by predicting user preferences through collaborative filtering', *International Journal of Robotics Research*, vol. 35, no. 13, pp. 1587–1608.

Breuer, T, Macedo, GR, Hartanto, R, Hochgeschwender, N, Holz, D, Hegger, F, Jin, Z, Mueller, CA, Paulus, J, Reckhaus, M, Ruiz, JA, Plöger, P, & Kraetzschmar, GK 2012, 'Johnny: An Autonomous Service Robot for Domestic Environments', *Journal of Intelligent & Robotic Systems*, 66, pp. 245-272.

Choi, JW, Oh, DI & Kim, SW 2006, *CPR localization using the RFID tag-floor*, vol 4099 LNAI, Lecture Notes in Computer Science, Springer Verlag.

Fausset, C, Kelly, A, Rogers, W & Fisk, A 2011, 'Challenges to Aging in Place: Understanding Home Maintenance Difficulties', *Journal of Housing for the Elderly*, vol. 25, no. 2, pp. 125–141.

Gallagher, G, Srinivasa, SS, Bagnell, JA, Ferguson, D 2009, 'GATMO: A Generalized Approach to Tracking Movable Objects', *IEEE International Conference on Robotics and Automation, Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, p. 2043.

Henning, M 2004, 'A new approach to object-oriented middleware' 2004, *IEEE Internet Computing, Internet Computing, IEEE, IEEE Internet Computing*, no. 1, p. 66.

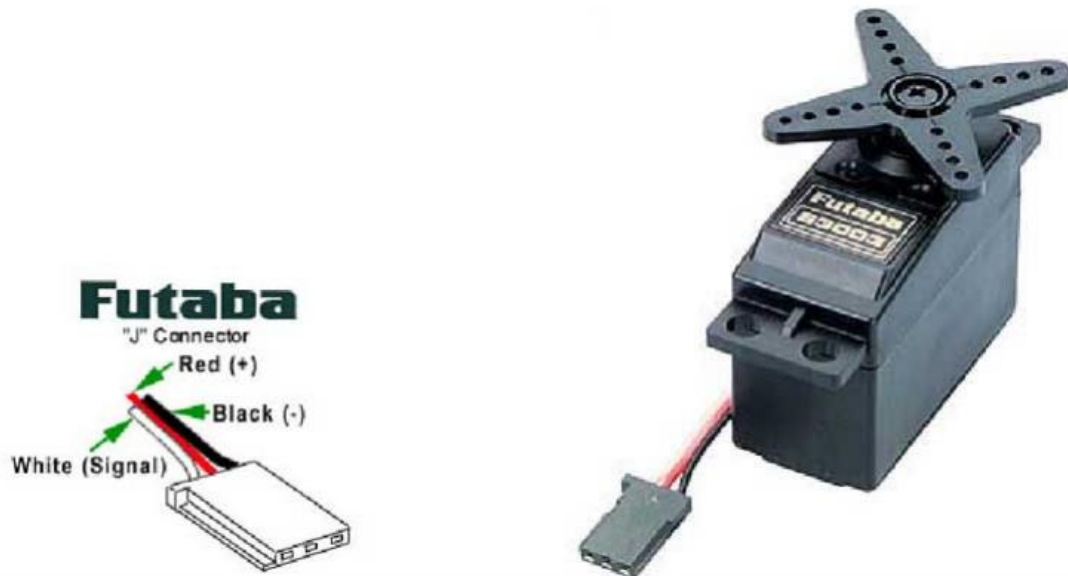
Kopacek, P 2016, 'Development Trends in Robotics', *IFAC PapersOnLine*, vol. 49, pp. 36–41.

Bibliography

- Kulkarni, S & Junghare, S 2013, 'Robot based indoor autonomous trash detection algorithm using ultrasonic sensors', *CARE 2013 - 2013 IEEE International Conference on Control, Automation, Robotics and Embedded Systems*, Proceedings.
- Pyo, Y, Cho, H, Jung, R & Lim, T 2017. *ROS Robot Programming*, ROBOTIS Co.,Ltd, pp.41-43.
- ROBOTIS e-Manual 2019, *TURTLEBOT3*. Available from:
<<http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>>. [5 March 19].
- Sridharan, K, Rajesh Kumar, P, Sudha, N & Vachhani, 2008, 'A novel CAM-based robotic indoor exploration algorithm and its area-efficient implementation', *IECON Proceedings (Industrial Electronics Conference)*, pp. 2419–2424.
- Srinivasa, SS, Ferguson, D, Helfrich, CJ, Berenson, D, Collet, A, Diankov, R, Gallagher, G, Hollinger, G, Kuffner, J & Weghe, MV 2010, 'HERB: A home exploring robotic butler', *Autonomous Robots*, vol. 28, no. 1, pp. 5–20.
- Youngkak, M, Seungwoo, K, Dongik, O & Youngwan, C 2008, 'A study on development of home mess-cleanup robot McBot', *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, AIM, pp. 114–119.

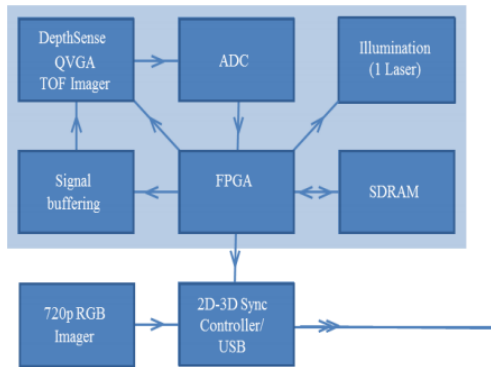
Appendix A: Datasheets

S3003 FUTABA SERVO



...S3003 FUTABA SERVO...			
Detailed Specifications			
Control System:	+Pulse Width Control 1520usec Neutral	Current Drain (4.8V):	7.2mA/idle
Required Pulse:	3-5 Volt Peak to Peak Square Wave	Current Drain (6.0V):	8mA/idle
Operating Voltage:	4.8-6.0 Volts	Direction:	Counter Clockwise/Pulse Traveling 1520- 1900usec
Operating Temperature Range:	-20 to +60 Degree C	Motor Type:	3 Pole Ferrite
Operating Speed (4.8V):	0.23sec/60 degrees at no load	Potentiometer Drive:	Indirect Drive
Operating Speed (6.0V):	0.19sec/60 degrees at no load	Bearing Type:	Plastic Bearing
Stall Torque (4.8V):	44 oz/in. (3.2kg.cm)	Gear Type:	All Nylon Gears
Stall Torque (6.0V):	56.8 oz/in. (4.1kg.cm)	Connector Wire Length:	12"
Operating Angle:	45 Deg. one side pulse traveling 400usec	Dimensions:	1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
360 Modifiable:	Yes	Weight:	1.3oz. (37.2g)

DS325 Specifications



TOF Sensor	QVGA-A (320x240)
FOV (HxV)	74x58
Illumination	Single Laser
Range	Short Range Only: 15cm-1m
RGB	
	Resolution HD 720p
	FOV (HxV) 74 x 58
CONNECTIVITY	USB 2.0
AMBIENT LIGHT	Typical indoor
TEMP RATING	10 – 40 °C (external air temperature)
RGBZ REGISTRATION	Yes
DIMENSIONS	10.5cm (W) x 3cm (H) x 2.3cm (D)
Drivers	PC Windows 7/8 PC Ubuntu ARM Android (April '14) ARM Linux (April '14)





TIDY UP FUNCTION DEVELOPMENT FOR SERVICE ROBOT


Problem Statement and Motivation

- Number of ageing adults increases, face difficulties when doing household chores.
- Working adults don't have time for regular household cleaning.
- Intelligent home service robots with lack of manipulating ability to interact with house objects.

Objectives

- To reduce human resource required for tidying up messy places.
- To provide additional function (tidy up) for home service robots.

Tools Used

	TurtleBot 3 Burger - ROS framework		Robot Arm - Object manipulator		Softkinetic ds325 - Object detection
---	--	---	--	---	--

Result



Able to recognise and move to toy object location, pick up object and place it at starting point.

Discussion

- TurtleBot meets the objective requirements. But weaknesses includes:
- Object detector does not show high accuracy since false detection can still occur.
 - Inconsistent bounding box size and location on detected object.
 - Fixed pick up point since robot arm cannot adjust its gripper position.
 - Lack of robust navigation method.

Conclusion

The robot is able to perform tidy up function to clear up a messy location with toy objects autonomously. This project provides a low-cost solution for users who wish to have an autonomous service robot to tidy up their household.

By Liew Zhao Ying
Made with PosterMyWall.com

Plagiarism Check Result

Chapter 1 Introduction

1.1 Problem Statement and Motivation

According to Fausset et al. (2011), ageing adults will face difficulties when it comes to home maintenance. The elderly might risk injuring themselves if they were to complete household chores, same applies to the disabled. With more working adults in the society, people will find it harder to make time for chores. Families with children will find it troublesome to tidy up children's toys. Besides that, without regular maintenance to keep the household clean and organised, it can be hard for the owner to keep track of their belongings. People must decide whether to sacrifice their off time to maintain their household cleanliness or hire a maid to keep their house clean and tidy. In the long run, these solutions will be costly in terms of time, energy, and money. However, with the growing trend of automation systems, robotics can be applied to replace human to complete tasks. When compared to humans, robots are more durable, productive, effective, and consistent when it comes to executing physical tasks. Robots can perform the given task well without having to prioritise its own safety and they are very flexible, making it possible for robots to work under most environments.

Intelligent home service robots can navigate the household environment and occasionally, move objects by themselves. One well-known example of home service robot

Match Overview

2%

- Submitted to University... <1% >
Student Paper
- Submitted to University... <1% >
Student Paper
- Submitted to Universiti ... <1% >
Student Paper
- Submitted to Universiti ... <1% >
Student Paper
- Submitted to University... <1% >
Student Paper
- Submitted to University... <1% >
Student Paper
- learn.komby.com <1% >
Internet Source
- Submitted to Middlese... <1% >
Student Paper
- www.scribd.com <1% >
Internet Source
- eprints.utar.edu.my <1% >
Internet Source

Plagiarism Check Result

Similarity Index		Similarity by Source	
2%		Internet Sources:	0%
		Publications:	0%
		Student Papers:	2%

[include quoted](#) [include bibliography](#) [excluding matches < 8 words](#) [download](#) [print](#)

mode:

<1% match (student papers from 07-Jun-2019)
[Submitted to University of Technology, Sydney on 2019-06-07](#)

<1% match (student papers from 08-Apr-2019)
[Submitted to Universiti Tunku Abdul Rahman on 2019-04-08](#)

<1% match (student papers from 20-Feb-2016)
[Submitted to Universiti Tenaga Nasional on 2016-02-20](#)

<1% match (student papers from 18-Apr-2016)
[Submitted to University of Liverpool on 2016-04-18](#)

<1% match (student papers from 11-May-2018)
[Submitted to University of Sheffield on 2018-05-11](#)

<1% match (student papers from 12-Nov-2010)
[Submitted to University of Northumbria at Newcastle on 2010-11-12](#)

<1% match (Internet from 10-Feb-2015)
<http://learn.komby.com>

<1% match (Internet from 19-Mar-2019)
<http://eprints.utar.edu.my>

<1% match (Internet from 14-Jul-2019)
<https://www.scribd.com/document/321764678/A-Study-on-Consumer-Perception-on-HDFC-Life-Insurance-Product>

<1% match (student papers from 10-May-2019)
[Submitted to Middlesex University on 2019-05-10](#)

<1% match (student papers from 08-May-2018)
[Submitted to University of Sheffield on 2018-05-08](#)

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	
ID Number(s)	
Programme / Course	
Title of Final Year Project	

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: _____ % Similarity by source Internet Sources: _____ % Publications: _____ % Student Papers: _____ %	
Number of individual sources listed of more than 3% similarity: _____	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Signature of Co-Supervisor

Name: _____

Name: _____

Date: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION & COMMUNICATION
TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	
Student Name	
Supervisor Name	

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. _____ (Signature of Student) Date:	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. _____ (Signature of Supervisor) Date:
--	--