School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Human-Robot Interaction for Cashier Robot

Rafael Papallas

Submitted in accordance with the requirements for the degree of BSc Computer Science (Industrial)

Session 2016/2017

The candidate confirms that the following have been submitted.

Items	Format	Recipient(s) and Date
Two printed reports	Report	SSO (09/05/2017)
Online report	PDF file	VLE (09/05/2017)
GitHub repository	Online (Appendix)	SSO & VLE (09/05/2017)
Demonstration videos	Video (Appendix)	SSO & VLE (09/05/2017)
Project Homepage	Website (Appendix)	SSO & VLE (09/05/2017)

Type of project: Exploratory Software (ESw)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) _____

© Session 2016/2017 The University of Leeds and Rafael Papallas

Summary

Baxter is a very friendly humanoid robot by Rethink Robotics. Baxter is all over the world; in factories, research laboratories and also at the School of Computing at the University of Leeds.

This work investigates several perception and manipulation algorithms to implement a cashier robot. The main objective of this project is to make Baxter a fully capable cashier that can accept payments in a proposed sweetshop and handle change.

In addition, this piece of work was integrated into last year's project. The aim of last year's project was to make Baxter operator of a sweetshop.

Acknowledgements

I would like to thank my supervisor Dr Mehmet Dogar who throughout the second semester have guided and helped me with several aspects of the project. During our weekly meetings, he listened to my ideas and plans and he gave invaluable suggestions and comments as well as alternative solutions when things did not turn to be as initially planned or thought.

In addition, I would also like to thank my assessor, Professor Anthony Cohn. His early feedback on Scoping and Planning document as well as his comments in the progress meeting earlier in the project, helped me to identify some tricky parts of the project as well as to make me aware of some difficulties I should expect from such a project, especially about my estimation of tasks that in some cases I have underestimated.

Furthermore, I would like to thank my family for supporting me during this project and their motivation they have given to me.

In addition, I would like to thank all of the participants that helped to test the end result.

Finally, I would like to thank Muhannad Al-Omari a PhD student at the School of Computing since in my first days in the robotics laboratory helped me to get familiar and also some feedback he gave mainly for the calibration tool.

Contents

Chapter 1

Introduction

1.1 Context

Baxter is a humanoid robot that is used all over the world; from factories to universities. The University of Leeds is no exception.

Last year, a student programmed Baxter to operate a sweetshop. Baxter was able to detect customers, receive orders and prepare the orders for the customers. The main concept behind last year's project is that the customer orders different coloured sweets. Baxter is able to distinguish the colours of the sweets and pick the ones that the customer have ordered.

In this project, the aim is to explore several ideas and algorithms for Human-Robot interaction with Baxter as the cashier. This project contributes to the overall vision of School of Computing with Baxter as the shopkeeper.

Baxter during the open days at the School of Computing welcomes new students and the shopkeeper demo will likely be used.

Figure 1.1: The Baxter Robot by Rethink Robotics Inc[1]

At the beginning of the project, a Scoping and Planning document have been submitted with the initial plan of the project. However, since the submission, some changes have occurred that are highlighted in appendix F.

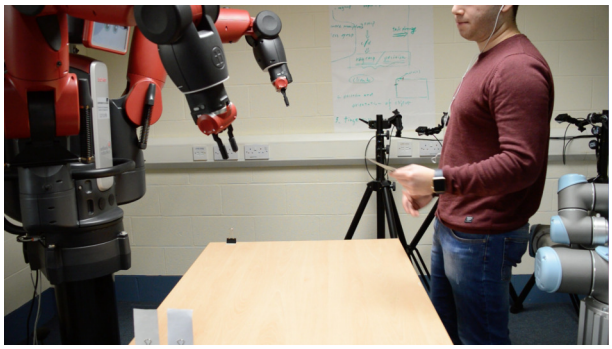
[1] Picture Reference: `http://www.hizook.com/blog/2012/09/18/baxter-robot-rethink-robotics-finally-unveiled`

Finally, in appendix C access to other deliverables of the project like videos, code repository and project's website can be found.

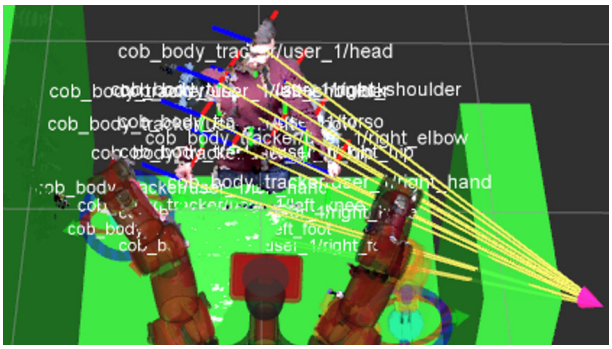
1.2 Project Aim

The aim of this project is to explore several ideas and algorithms for Human-Robot interaction and collaboration with the aim of making Baxter a capable cashier. In specific, this project will explore how Baxter can handle payments in a proposed sweetshop. The robot should be able to receive payments, work out the amount given by the customer and handle change if needed.

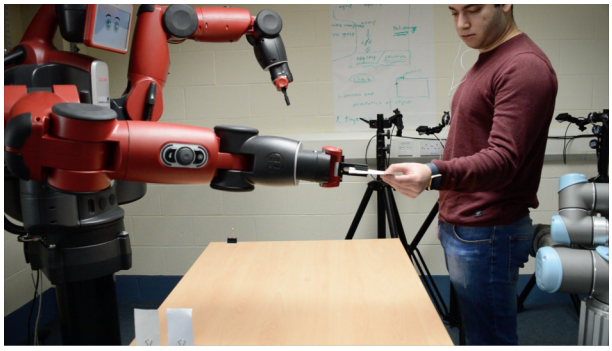
Moreover, last year's project will be integrated with this project to create a unified demo. Therefore the project needs to be designed and implemented in such a way that will make easy the integration with the last year's project at the end.



(a) User and Baxter in scene. User gets ready to perform the handover task.



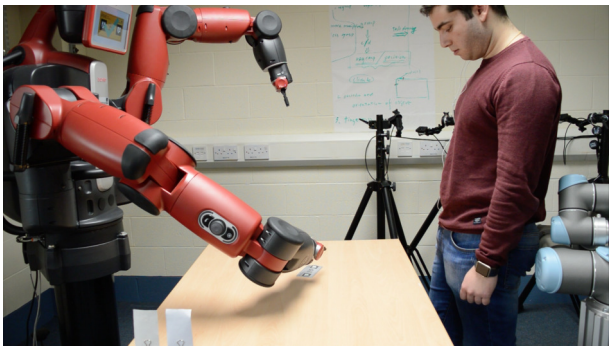
(b) Skeleton Tracker in Rviz showing how the software detects the positioning of user's body.



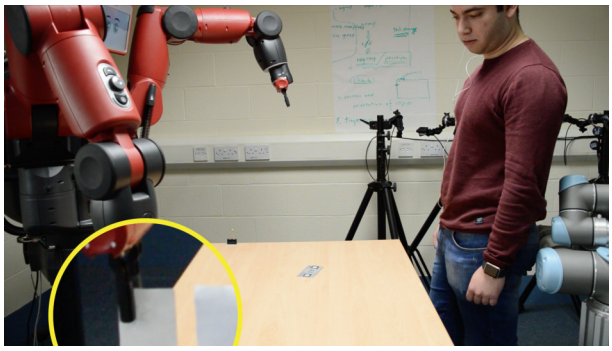
(c) Banknote handover from user's left hand based on the position identified by the skeleton tracker earlier.

(d) Money recognition state. Baxter brings the banknote to his forehead camera to perform the money recognition task.

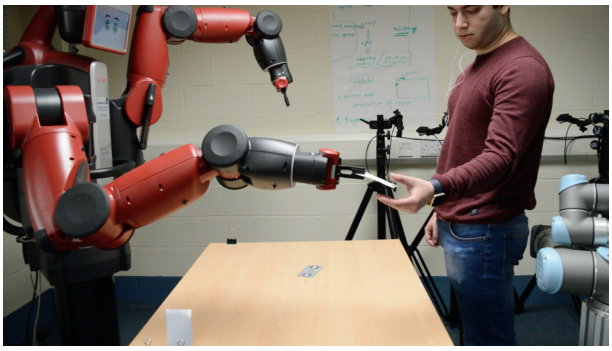
(e) Recognition indication message. Baxter shows that the recognition was successful.



(f) Baxter leaving banknote on the table after the successful recognition.



(g) Baxter grasping change from the table.



(h) Baxter returns change to user's hand.

Figure 1.2: Project in pictures

1.3 Problem Domain

The main problem to be tackled is how to detect a correct hand-pose signal of the user. Once this signal is detected the robot needs to plan a path to pick the banknote from the customer's hand (As you can see from Figure 1.2a to Figure 1.2c).

Another problem to be solved is how to recognise and distinguish different banknotes and to handle corner cases when a customer gives no money or invalid banknotes (As shown from Figure 1.2d to Figure 1.2e). A real cashier can distinguish a five pound from a ten pound. Baxter should perform the same activity but with made-up banknotes.

Baxter will need to handle change (Figure 1.2g and Figure 1.2h). In the case where the customer's payment is more than the grand total, the robot needs to work out the correct amount of change and return the change to the user or end the interaction if the customer gave the exact amount.

The project have some risks that are highlighted in the risk assessment under appendix E.

1.4 Project Objectives

Given the above problem domain the following objectives have been conducted:

1. Understand and gain experience on fundamental robotics theory including robotics in general, kinematics, simulator environment, motion planning, camera pose calibration and human-robot interaction.
2. Understand and gain experience using ROS[2] and Python to interact and program Baxter, including other tools used in robotics like Rviz.
3. Develop hand-pose detection library using an RGB-D camera or Motion Capture System.
4. Develop a picking algorithm to pick banknote from user's hand and a giving algorithm to return change to user's hand.
5. Develop computer vision algorithm to recognise different banknotes.
6. Develop the logic for change handling.

[2]Robot Operating System

Chapter 2

Literature Review & Background Research

2.1 Robot Operating System (ROS)

Robot Operating System (ROS for short) is a middleware software running on a variety of operating systems including GNU/Linux, macOS and Windows. ROS provides several robotic libraries and tools that simplifies the process of developing software for robots [1].

ROS has several active versions including ROS Indigo Igloo which is the one used for the purposes of this project. ROS Indigo Igloo targets Ubuntu 14.04 (LTS) [2] and is the Ubuntu version running in robotics laboratory. Moreover, ROS is an open-source project with a great community behind it.

2.2 Simulation Environment (Gazebo)

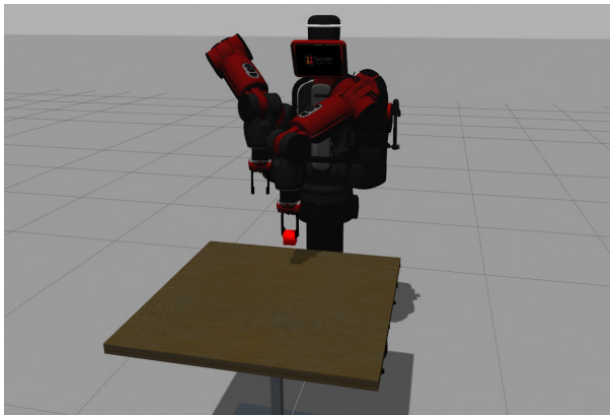


Figure 2.1: Gazebo Simulator with Baxter Robot[1]

A great ROS feature is that it is compatible with a simulation environment called Gazebo. Gazebo allows the user to simulate the robot and run processes and scripts on the simulator as opposed to the real robot.

The use of simulator makes very efficient the process of testing code since (1) it does not require to be connected to the robot hardware and (2) the development and testing of the software in the simulator is location-independent.

[1] Picture Reference: `http://sdk.rethinkrobotics.com/wiki/Baxter_Simulator`

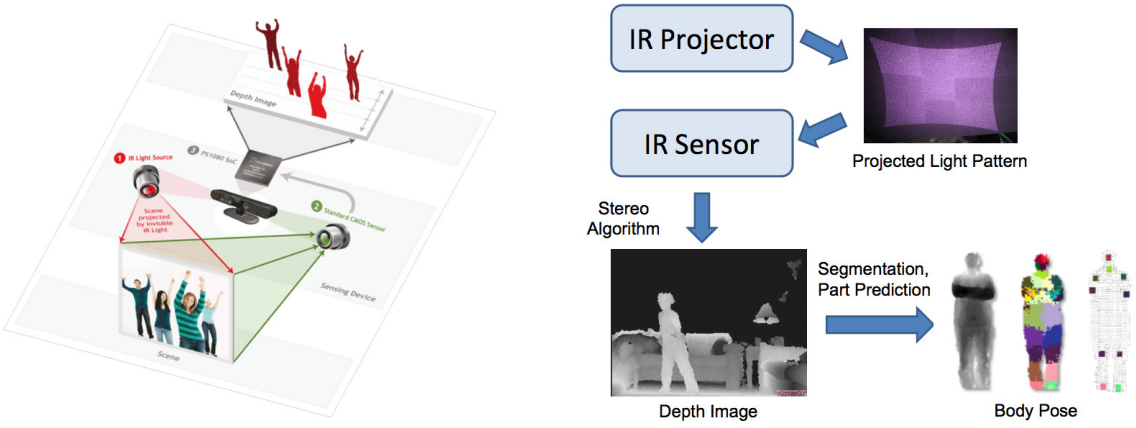
Additionally, Gazebo also provides real-life objects like a desk, bowl and other everyday objects which they can be added in the simulation environment.

2.3 Camera Sensor

Some of the project's tasks require the use of camera sensor. There are two notable options for the sensor. The use of: (1) an RGB-D camera (Kinect-like sensors) or (2) a Motion Capture System. School of Computing provides both options.

2.3.1 RGB-D Camera

An RGB-D camera sensor (or depth sensor) is able to record RGB frames with depth information per pixel [3]. This technology captures the 3D space with the sense of x, y and z-axis.



(a) Depth sensor overview

(b) Depth sensor step-by-step

Figure 2.2: Depth sensor overview (Image taken from: [4])

As shown in Figure 2.2a, the projector of the RGB-D camera (red arrows) sends invisible IR light that is then bounded back to the CMOS sensor (green arrows). The camera's driver will work out and produce the depth image which in turn will define how far objects or people are from the camera [4].

The Figure 2.2b shows the step-by-step process. The IR Projector creates the "Projected Light Pattern" which is then bounded back to the IR Sensor. Then a Stereo Algorithm is used to produce the depth image, which then using segmentation and part prediction to produce the body pose [4].

2.3.2 Motion Capture System

Another option for camera sensor is the Motion Capture System (MCS for short).

A motion capture system requires the potential users to have recognisable markers on their body [5]. For example, in the case of this project, a glove with markers on both user's hands. Furthermore, in most of the cases where Motion Capture System is used, a number of these camera sensors are needed to capture the full scene.

Although this system is very accurate, the requirement of markers to do the detection is not very attractive in the project's context since it will require every user to wear gloves with markers.

2.3.3 Related work

In [6] Ramey et al. have integrated Microsoft Kinect in a social robot for gesture recognition. Their motivation is to capture non-verbal gestures that are sent by users so the robot can better understand non-verbal information. The use of RGB-D sensor allowed them to implement the gesture recognition with success. It is worth mentioning that Ramey et al. supports that the use of IR depth sensor that is provided by the Microsoft Kinect, and RGB-D sensors in general, is low-light-independent. This means that light conditions will not affect the accuracy of the solution [6]. This property of light condition is a desired property for the purposes of this project.

Ott et al. [7] describes their work on a motion recognition using motion capture system and markers. Specifically, the motivation of the project is to imitate human motions by a humanoid robot through the capture system and using marker point measurements. This work confirms that motion capture system can be used in the context of robotics for human-robot interaction.

Finally, Cheng et al. in [8] highlight their work on a friendly and interactive robot system. The motivation of their work is to provide a communication protocol between non-expert users and robots through natural gesture recognition [8]. They also describe that most of the robots require mouse and keyboard which is not easy to operate by non-expert users. To achieve the end-result Cheng et al. uses a Microsoft Kinect (RGB-D) sensor to capture the skeleton of the users and translate their movements into gestures. This work again, shows how easy, accurate and affordable an RGB-D sensor is in the context of human-robot interaction, mainly because it requires no further configuration, it can work in low-light conditions and can accurately track different body parts. All these properties are the ones required for the purposes of this project.

2.3.4 Comparison of the two sensor technologies

Motion Capture Systems although very accurate requires the potential user to wear marker points on their hand. In addition, Motion Capture Systems requires significantly more time to configure than RGB-D sensor due to the fact that marker points needs to be attached to the user. Moreover according to Corazza et al. the markers in some cases can influence user's movement and also requires a controlled environment to be configured [5].

Unlike Motion Capture Systems, RGB-D cameras do not require further configuration or user to wear marker points. Therefore RBG-D sensors are considered more autonomous in the sense that they do not require further configuration other than the software that analyses and process the picture frames recorded. Unfortunately, RGB-D cameras can be considered noisy and they may track unwanted items [9]. However, in [9], Huang et al. have applied a filtering technique to reduce noise from an RGB-D camera using "Exponentially Weighted Moving Average" [9] which may be considered in case RGB-D camera turns to be too noisy.

In addition, Zhang, a Principal Researcher and Research Manager at Microsoft Research in his publication "Microsoft Kinect Sensor and Its Effect" states the following: "Simply put, skeletal tracking must ideally work for every person on the planet, in every household, without any calibration." [10]. This comment, coming from a Microsoft researcher, confirms that skeleton tracking can work with any person in any environment using RGB-D sensors. This is very positive since the use of an RGB-D sensor in this project is specifically to integrate a skeleton tracker for hand-pose detection.

With all this said, an RGB-D camera is probably the best first choice for the project due to the fact that is more autonomous than a Motion Capture System, however, MCS will be considered if RGB-D turns to be inaccurate.

2.4 Skeleton Tracking

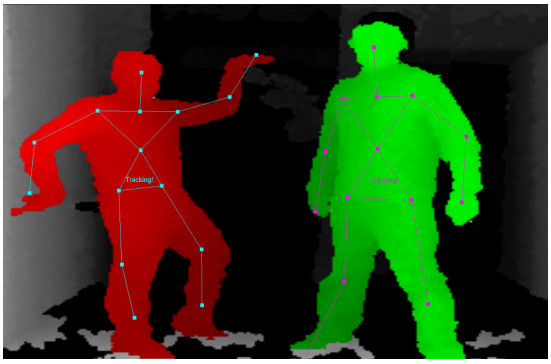
Skeleton tracker is the software that uses a camera sensor to track the pose of user's body.

As can be seen from Figure 2.3a, skeleton trackers are able to track multiple users at the same time. Major body parts like head, left/right hand, torso, left/right foot are being tracked by the skeleton tracker.

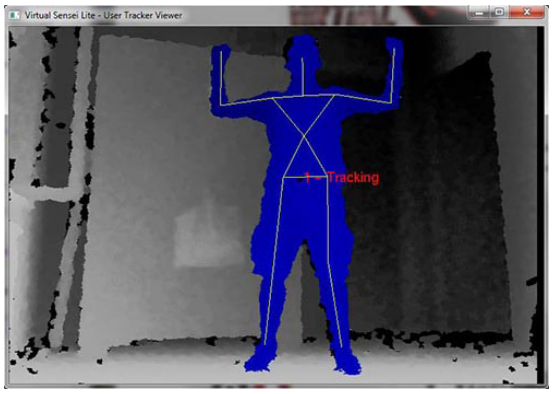
Usually, Skeleton Trackers publish those parts as `tfs`[2] in real time. A very popular skeleton tracker among the robotics community is the `openni_tracker`[3].

[2] A `tf` stands for transformation and it is a point in space defined by x, y and z values.
[3] `http://wiki.ros.org/openni_tracker`

The `openni_tracker` publishes tf's for every body part of the user. One of the disadvantages of skeleton trackers is that they require a full view of the user and in some cases will fail to detect the skeleton if the full body of the user is not visible to the camera. This is a problem for the project since the camera would not be able to be mounted on Baxter since Baxter will have a table in front of him that will block the camera view.



(a) Skeleton Tracker example[4]



(b) The psi pose needed to detect a user (forming the Greek letter psi Ψ)[5]

Figure 2.3: Skeleton Tracker example including the psi pose

In addition, `openni_tracker` in specific requires the user to do the "psi" pose, which requires the user to raise both of their hands above their head and bent to ninety degrees (see Figure 2.3b). This will be awkward to explain to users that will interact with Baxter however. Other libraries do not require the user to do the "psi" pose, like the `cob_people_detection`[6] library.

Both libraries mentioned will be tested and the best performing, for the purposes of the project, will be selected. Given that the later library does not require the "psi" pose as a requirement to detect a user it will probably be the first tracker to be tested.

2.5 Hand-Pose Detection & Human-Robot Handover

Another major task that needs to be explored is how to detect a correct hand pose signal over false positives hand-poses.

2.5.1 Related work

Huang et al. in [9] describe a robot and human working together to unload dishes from a dish rack performing a human-robot handover task. This requires the robot to slow down

[5] Picture Reference: `http://www.pointclouds.org/assets/uploads/gsoc14-people.png`
[5] Picture Reference: `http://www.virtualsensei.it/lite/help/`
[6] `http://wiki.ros.org/cob_people_detection`

or even to stop if the human is busy trying to load the dish to a shelf. In addition, they use a common reduction technique for Kinect-like sensors called "Exponentially Weighted Moving Average" to reduce noise from the sensor. Although the work is impressive, is significantly complicated and advanced for the tasks required for this project, since the handover process in [9] is not deterministic, where in this project the handover will be deterministic after a sequence of events[7]. Therefore in this context, the exact time where the hand-pose should be detected will always be deterministic after a series of events happened as opposed to waiting for the user to become available as like what is described in [9].

Another work is the one by Micelli et al. in [11].

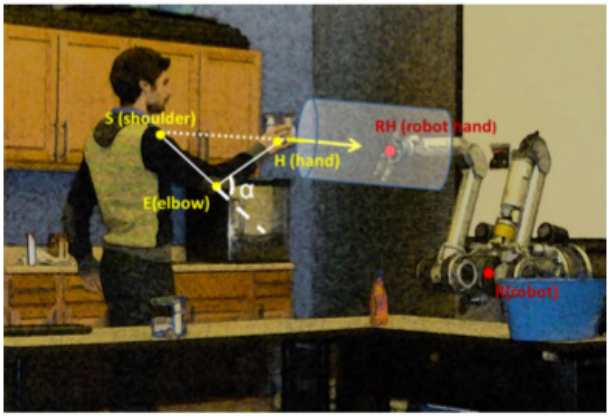


Figure 2.4: Detecting hand-pose (Image taken from: [11])

In [11] the authors among others highlights an interesting idea about how they detect a true hand-off signal. In specific the following needs to be true to identify the hand-pose:

- The subject is close to the robot.
- The vector of humans' shoulder and hand points are aligned with the robot's hand.
- The angle of humans' elbow is bent to a fixed value α.
- The subject is in this pose for five consecutive frames, to avoid false positives detections.

These seem to be good observations and they might be used in this project to identify hand-poses. Their observations, however, depends on the skeleton tracker of choice. The skeleton tracker needs to detect body parts like shoulder and elbow.

2.5.2 Implementation ideas

An interesting observation made when dealing with real cashiers is that the cashiers' effort is only made when receiving money (i.e when the customer is paying) since requires to

[7] Change return will happen immediately after user gave money to Baxter

find the hand-pose and pull the money from the customer's hand. In contrast, when the cashier returns the money back (i.e cashier is giving change), cashier only move their hand in approximation towards the customer but is the customer's effort to pull the money from the cashier's hand. This means that the task to receive money from the customer is more complicated for the cashier but the inverse is more straightforward. Therefore, when Baxter needs to pull money, will require different algorithms. When the cashier returns money has to simply predict an approximation of the hand-pose that will make the customer to move their hand there and pull the money from the cashier.

A reasonable implementation would be the following:

- Using a skeleton tracker detect user's hand pose and ensure this is a true hand-off signal by waiting for at least five consecutive frames [11].
- Once the pose is detected check whether the pose is within a given area, above and within the table in front of Baxter.
- Record this initial hand-pose so it can be used later.
- Reach the user to the hand-pose detected to retrieve the money.
- When Baxter returns change back to the customer, the initially recorded hand-pose can be used to approach the user, as like what happens in a real cashier-customer scenario where the cashier gives the change to an approximation towards to the customer.

2.6 Understanding Baxter Robot by Rethink Robotics

Baxter is a very sophisticated and complex robot. In this section, general information will be explored to understand how Baxter is designed but also study related work and how Baxter is used to interact with humans.

2.6.1 Related work

Kaipa et al. [12] presented their work on Baxter for safe and efficient human-robot interaction and collaboration. Baxter in this context is used in industrial assembly jobs. Baxter is programmed to pick parts from a bin (bin-picking) and assembly them. Human acts as a supervisor that supervises the process. In the case where the robot's perception system made a mistake then the human supervisor, using a laser, indicates to the robot which part to pick instead [12]. Baxter is aware of the human supervisor and the authors have implemented a safety mechanism that makes Baxter obstacle-aware. In specific, the robot will pause when a collision between the robot and the user is detected. The interesting

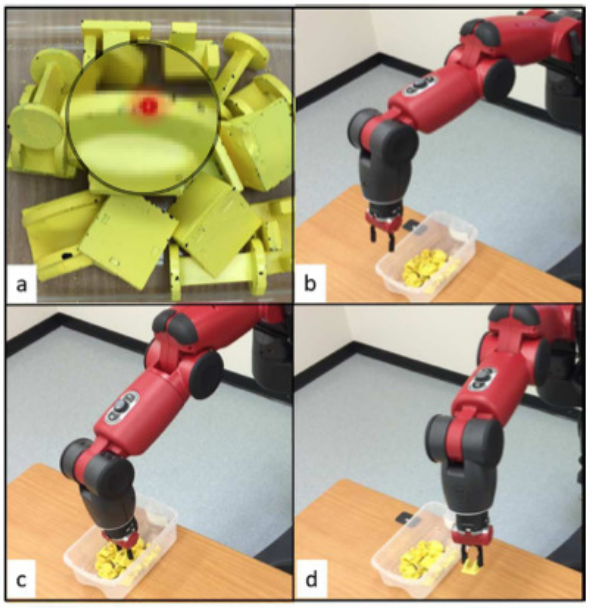


Figure 2.5: Baxter picking (laser) indicated part from the bin (Image taken from: [12])

part is that this safety feature called stop-go conforms to the 10218 ISO [12]. As can be seen from Figure 2.5(a) the red dot is the laser indication by the user. The robot reads the input from the user and it picks the indicated part.

Another notable and related work is the one in [13]. The authors explored a human-robot interaction for body motion tracking on Baxter. In this paper, the authors use an RGB-D sensor again (Kinect Xbox 360) to track the skeleton of the user. They have tried two different methods: (1) inverse kinematics and (2) vector approach [13]. Python and RosPy have been used to program Baxter. The authors highlight the importance of choosing a good skeleton tracker software. In specific they list the following requirements for choosing a skeleton tracker [13]:

- Extracting skeletal data should be possible.
- Should be compatible with multiple operating systems including Linux.
- Should be well documented.
- Simple to use and have effective results.

Although some of the requirements are reasonable, in the context of this project multiple operating system support and extracting skeletal data are not required. With that said, the requirements for skeleton tracker in this project is that it needs to be simple to use and well documented.

The authors in [13] after observing some results they have listed some advantages and disadvantages of using vector approach over the inverse kinematics one. Specifically, Reddivari et al. [13] state that the vector approach can be used to replicate human motion

where inverse kinematics can not replicate **exactly** human motion, but inverse kinematics can be used in picking and placing jobs. This is positive since in this project the use of inverse kinematics will be used once the hand-pose is detected.

2.6.2 Arm Joints and End-effectors

Baxter has seven degree-of-freedom joints on each arm [14]:

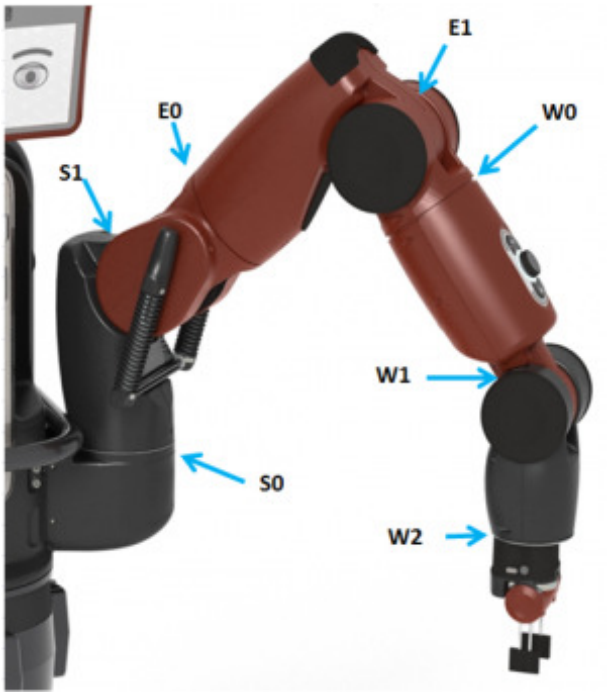


Figure 2.6: Baxter's arms joints (Image taken from: [14])

Each joint is named uniquely as can be seen from Figure 2.6. To distinguish the two hands the `left` or `right` prefix is appended to each joint name to uniquely identify the joint on either Baxter's hand.

`S1`, `E1` and `W1` are the bend joints that allows Baxter's arm to bend into some minimum and maximum degrees where `S0`, `E0`, `W0` and `W2` are the twist joints that they also have their upper and lower bounds that are highlighted in Table 2.1.

Baxter has two end-effectors (`left_gripper` and `right_gripper`). A closed gripper has the value `0.0` and an open gripper the value `100.0`.

Finally, Baxter's head is also movable part, especially useful when interacting with users to make Baxter more interactive as well as to change the point of view of the forehead camera (see Section 2.6.3).

Table 2.1: Joints minimum and maximum values [14]

Joint	**Minimum**	**Maximum**	**Range**
Twist Joints			
S0	-1.7016	+1.7016	3.4033
E0	-3.0541	+3.0541	6.1083
W0	-3.059	+3.059	6.117
W2	-3.059	+3.059	6.117
Bend Joints			
S1	-2.147	+1.047	3.194
E1	-0.05	+2.618	2.67
W1	-1.5707	+2.094	3.6647

2.6.3 Baxter's Cameras

In addition, Baxter has three cameras; two located at each hand, exactly underneath the end-effector and one on his forehead.

One limitation when it comes to Baxter's built-in cameras is that due to USB system limitations only two can be activated and used at the same time. With that said, one of the cameras should be manually turned-off when not in use [15].

These cameras are not depth sensors but colour cameras. Such cameras can become handy for colour recognition. Hence, these cameras can be used for the money recognition part of the project.

2.6.4 SDK API

Rethink Robotics provides a very well documented API in both `C++` and `Python` programming languages with ROS libraries `roscpp` and `rospy` respectively.

These SDKs allows third-party developers and researchers to interact with Baxter and program Baxter. The SDK among others, provides Inverse Kinematics solver, gripper management (that is, it allows the grippers to open or close) as well as other useful functions.

For the purposes of this project, Python (rospy) will be used. Last year's codebase is built on Python and since the project would be integrated into last year's project, Python is clearly the only option here.

2.7 Manipulation Planning

In this project, manipulation planning refers to the task of getting or giving money from/to the customer. This is a complex problem for various reasons.

The main problem here is given an arbitrary point in space, to move Baxter's arm there. In Robotics the common way to solve this problem is the use of Inverse Kinematics solvers.

The problem that needs to be considered in here is that the object to be grasped is within the hands of the customer. That means Baxter will have to complete a handover task, which involves manipulation planning.

2.7.1 Kinematics

Craig define Kinematics as: "the science of motion which treats motion without regard to the forces that cause it. Within the science of kinematics one studies the position, velocity, acceleration and all higher order derivatives of the position variables." [16]

Forward Kinematics: Forward Kinematics is the deterministic solution to the following question: "Given all the joint configurations of the robot, what is the exact pose of the end-effector?" This question, states that if the angles of each joint of the robot are known, what is the position and rotation of the end-effector? Forward Kinematics can become helpful when Baxter will need to determine a specific pose of the first banknote on the table. Banknotes will be placed on the table so that the robot can give change back to the customer. At the beginning of the demo, the operator will probably be prompted to manually move Baxter's arm to the position of the first banknote on the table, and then Baxter will need to work out the pose of the current configuration. In this scenario, forward kinematics will be used.

Inverse Kinematics: Inverse Kinematics solves the following problem: "Given the desired posture and orientation of the goal, how can the joint configurations be calculated to reach this goal pose?" [16]. In other words, IK (Inverse Kinematics) will return a list of possible angles for each joint of the robot that when apply, the end effector will reach the pose in question. Usually, IK solvers return multiple solutions, as a pose can be reached by multiple joint configurations. Inverse Kinematics will be one of the core aspects of this project since there will be several times that Baxter will need to move his hand to the desired location. For example, when skeleton tracker identifies a hand-pose, Baxter will require solving this pose using IK solver to reach the pose, or when Baxter will need to grasp change from the table will again need, given a pose, to reach that pose.

2.7.2 Planners

Another important part of the manipulation process is the planner that is used. A planner is a software that does the manipulation, given the solution configuration of the joints.

Some planners provide collision-aware planning. Collision-aware planning means that the planners can avoid obstacles when doing the planning. Unfortunately, Baxter's default planner does not provide such feature out of the box.

Trajectory: Trajectory in Robotics is the timed series of position velocity and acceleration values for each joint of the robot [16]. These trajectories together make the motion of the movable part. A planner will produce trajectories as part of the manipulation process. Collision-aware planners will consider objects or obstacles while generating those trajectories so the planner can plan a path that does not collide with the obstacles.

MoveIt!

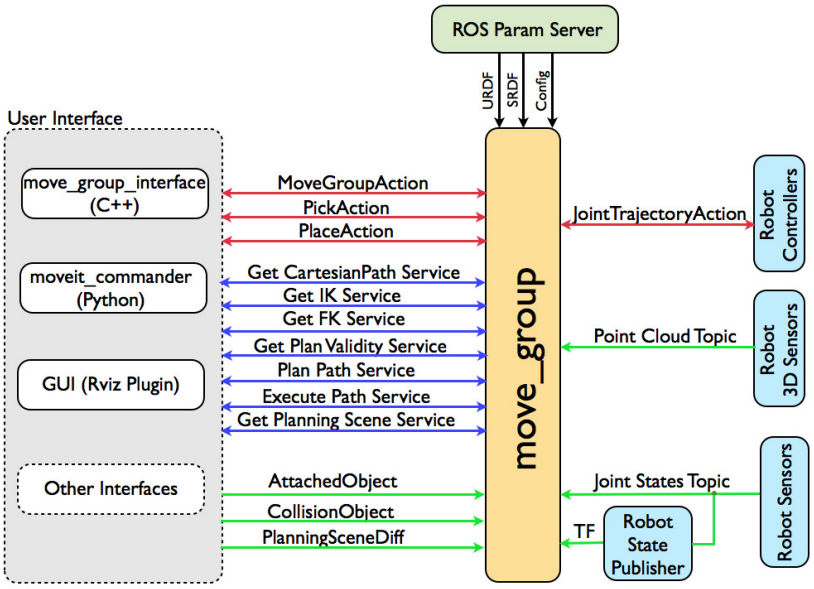


Figure 2.7: MoveIt! Architecture (Image taken from: [17])

MoveIt is a motion planning framework with 3D perception, kinematics and more importantly collision-aware planners [18].

Rethink Robotics and Baxter are fully compatible with MoveIt [19]. MoveIt! provides support to publish and create obstacles that planner can use during the trajectory generation to avoid those obstacles.

Given that the project might need to use collision-aware planner and that Baxter's default planner does not provide this functionality, the use of such framework like MoveIt! might be used.

Figure 2.7 shows an abstract image of "MoveIt!" architecture. MoveIt! provides a variety of interfaces for languages like `C++` and `Python` as well as a plugin for Rviz. These interfaces communicate with `move_group` as shown in Figure 2.7. `move_group` is mainly a movable part of the robot like Baxter's arm.

2.8 Money Recognition

Money recognition part is an equally complicated problem as the pose detection, given that it needs to be accurate in this context. A number of possible solutions have been conducted.

Banknote's colour can be used to identify the different notes. For example, a five pound note will be in orange and one banknote in blue. Identifying different kind of money will require distinguishing the banknote's colour.

Alternatively, since each different banknote will defer at least to some points (like the number printed on it), this information can be used to perform pattern-matching or template-matching.

Finally, another possible idea is to attach QR code on the banknote and using a QR code reader the algorithm can distinguish the different banknotes.

Each of these ideas provides different advantages and disadvantages. Each of the three solutions will be explored and tested so the best solution can be conducted.

Chapter 3

Perception Development

3.1 Skeleton Tracking

Two skeleton tracking libraries were discussed in Section 2.4. The library `cob_openni2_tracker` was chosen as the final library since it does not require the "psy" pose to detect the user. Moreover, this specific library it was the only one that worked with the camera sensor and it was very accurate during the testing. Other libraries including `openni_tracker` they either did not worked with the sensor (library did not published transformation for the parts) or they would not start at all. The RGB-D camera sensor used was "ASUS Xtion Pro Live".

3.1.1 Structure and technical details

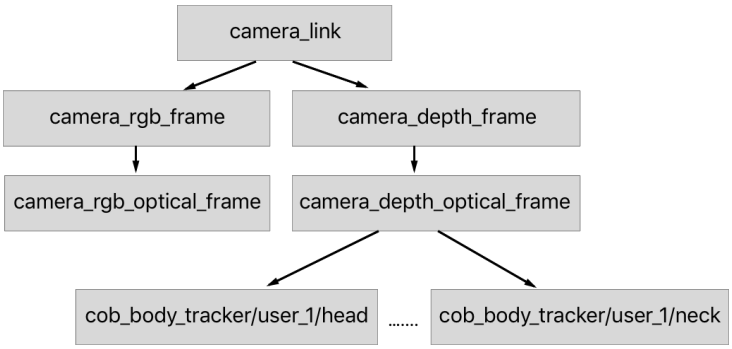


Figure 3.1: Skeleton tracker structure

Figure 3.1 shows the tree-like structure of the skeleton tracker. The last level of the hierarchy in the diagram is the actual body parts that are published by the skeleton tracker which includes: `left_hand` and `right_hand` among other body parts.

All these are children of the `camera_depth_optical_frame` and to distinguish different users each body part has the `user_n` as a prefix since `cob_openni2_tracker` supports multi-user tracking.

The skeleton tracker library runs in the background and publishes the user's parts in real-time. When the project requires access to body parts it calls the custom script `body_tracker_listener.py` that does a lookup for user's left or right hand. It will then return back the transformation of the user's hand (x, y and z) with a defined rotation.

The rotation was manually defined since Baxter's end-effector during the hand-off activity is required to be in a specific rotation.

Left Rotation = [-0.513, 0.520, -0.499, 0.467]
Right Rotation = [0.559, -0.504, 0.480, -0.451]

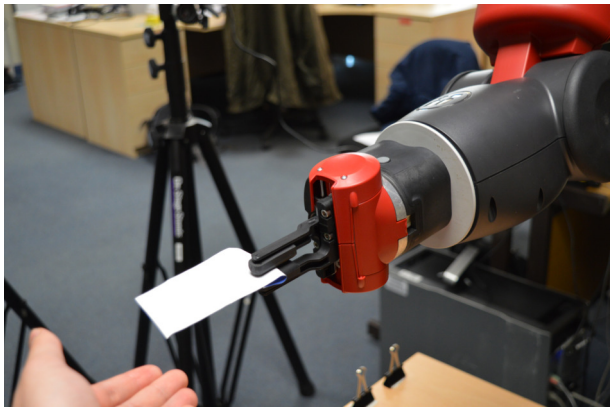


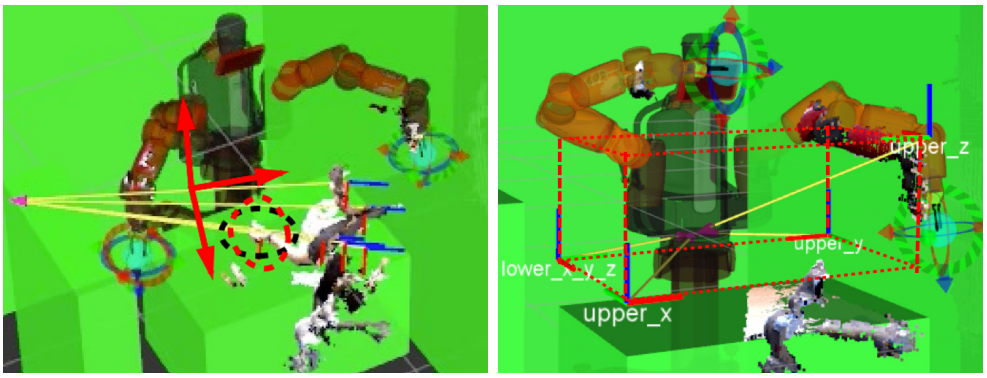
Figure 3.2: End-effector static rotation imitating true hand-off

3.1.2 Pose Elimination

It would be very inefficient to accept all the postures read by the skeleton tracker and pass them to the planner regardless if the pose is reachable or not. It is also inconvenient from the user's point of view to trigger the Robot whenever is moving his/her hand anywhere. When the basic solution was implemented, there was no such mechanism, and the detection time was taking from 30 seconds up to a minute in some cases, which was very slow. After a detailed investigation, the reason of being so slow was due to the fact that every pose (whether was relevant or not) was passed to the planner which in turn was trying to find a solution for every pose passed in. The planner took some seconds per pose, and hence the detection was very slow.

For this reason, a two-way pose elimination has been implemented to filter or eliminate the poses read by the skeleton tracker and only return the one that seems to be a true hand-off. With the help of this filtering technique, the detection time was dramatically reduced to less than 5 seconds on average.

The first way of the mechanism is to determine if a hand-pose was within some boundaries for at least two consecutive frames. This is achieved by recording the pose of the first hand-pose, apply some threshold of +/- 10cm to each axis (x, y and z) and use these boundaries to check consecutive frames to determine if they are in the same hand pose. The need of the threshold was added because skeleton trackers are not always very accurate and from one frame to another, the pose can defer. When the algorithm detects that at least two consecutive frames were within the boundaries, it will count the pose as a valid one and



(a) Determine if consecutive hand-poses is within the boundaries (boundaries = two circles)

(b) Four static `tf` showing the reachable area bounds; Red box showing the reachable area

Figure 3.3: Two-way pose elimination mechanism

will return it back to the main program. In Figure 3.3a the boundaries are shown with the two circles around the customer's hand.

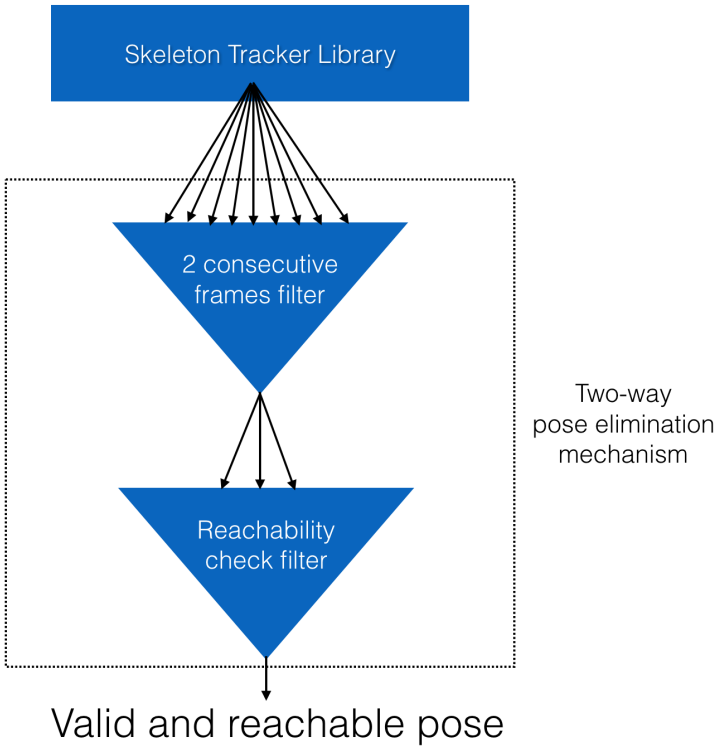


Figure 3.4: Visualisation of the two-way pose elimination mechanism

The second check in the two-way pose elimination is to check if the returned pose is within Baxter's reachable area. As can be seen from Figure 3.3b static transformations have been used to visualise the reachable area (`lower_x_y_z`, `upper_x`, `upper_y` and `upper_z`). With these four transformations, upper and lower bounds for each x, y and z-axis have been conducted. Having upper and lower boundaries for each x, y and z make easy to do a boundary check for the given pose and to determine if it is within the

reachable area or not. The reachable area was decided to be exactly above the table (just a few centimetres above the surface to avoid triggering false positives).

Figure 3.4 visualise the efficiency of the two-way pose elimination. As you can see from the figure, the skeleton tracker sends many poses per second. Many of those poses are not relevant at all. The first filter, filters the hand poses that are in the same position for at least two consecutive frames. The output of this filter is then used as the input of the second filter which checks the reachability of those hand poses. The output of the second filter means that the hand pose is intended and within the reachable area, which means that is likely a true hand off signal.

The reachability check, therefore, is now "MoveIt!"-independent and very efficiently the script can determine whether a posture is reachable or not without having to run through planners and their expensive computations.

3.2 Camera Calibration

The RGB-D camera used to perform the skeleton tracking task needs to be placed far away from Baxter so it can have a full view of the user. However, this create a problem that is discussed in this section.

3.2.1 Problem

The camera calibration in this report refers to the process of statically transforming the image view of the RGB-D camera to Baxter's base so the poses published by the skeleton tracker from the RGB-D sensor are relative to Baxter's position.

Since a table is placed between Baxter and the user, the RGB-D camera that tracks the human skeleton needs to be far away to have a clear view of the full human skeleton (because the table will block the view if the camera is mounted on Baxter for example). The skeleton tracker library, `cob_openni2_tracker`, knows nothing about this distance mismatch and hence the `tf` that will be broadcasted will not be relative to Baxter's position.

Relative positioning means that the camera will record the position of user's hand based on the distance the human is from the camera and not from the Robot, hence Baxter's IK[1] solver will not be able to find a solution.

[1] Inverse Kinematics

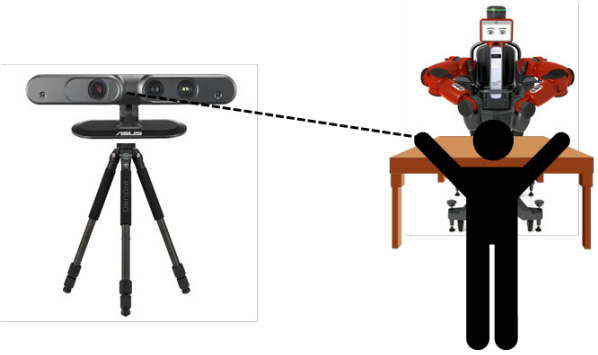


Figure 3.5: Distance mismatch problem

3.2.2 Solution

In the past, Muhannad Al-Omari a PhD student at the University have implemented a calibration tool[2] for a similar problem that allowed the user to calibrate the image of the RGB-D camera with the positioning of the robot through a user interface.

Based on Muhannad's idea, a forked version has been created to implement the idea for the purposes of this project. This modified version uses ROS topics relevant to project's RGB-D camera as well as implementing some new functionalities to make the process easier.

As you can see from the Figure 3.6, the tool provides a graphical user interface to change the positioning of the camera's base both in terms of transformation and rotation.

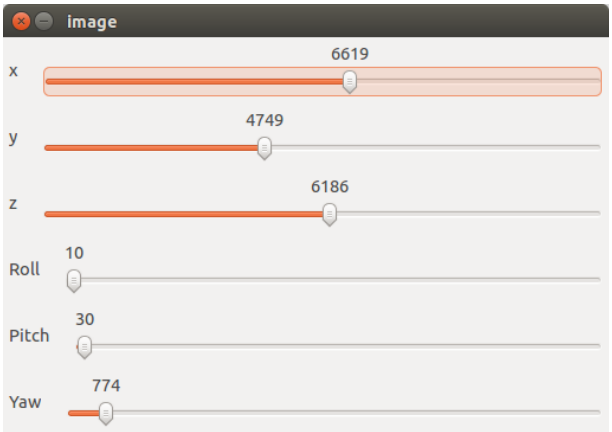


Figure 3.6: Camera Calibrator Tool - User Interface. (Cropped Screenshot[3])

How the end result is achieved: This solution requires the use of Rviz. Rviz provides a PointCloud functionality that allows to visualise the camera's frames in the 3D space. Moreover, the robot model needs to also be added to Rviz.

[2] Script URL: `https://github.com/OMARI1988/baxter_pykdl/blob/master/scripts/kinect_frame.py`

[3] Full screenshot: `https://github.com/papallas/baxter_cashier/wiki/Camera-Calibration`

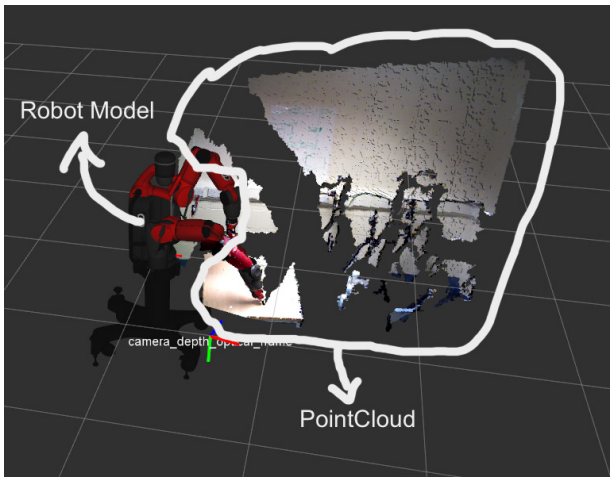
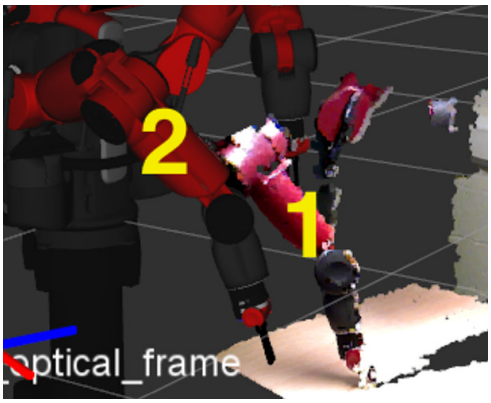
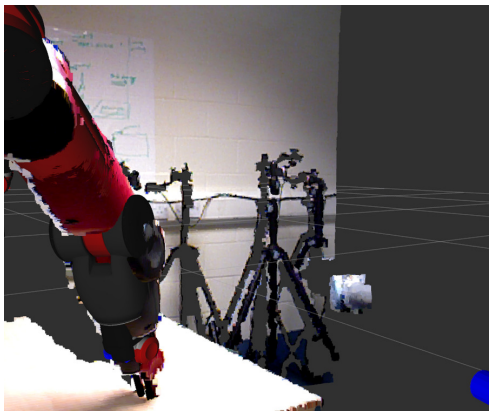


Figure 3.7: Rviz: Baxter Model and PointCloud2

The target is to bring the PointCloud2 image exactly above the robot model. When the PointCloud's view is correctly aligned using the interface mentioned in Figure 3.6 then Baxter and the camera are calibrated.



(a) Baxter not calibrated correctly



(b) Baxter Calibrated (1)

Figure 3.8: Baxter Calibration

As you can see from the Figure 3.8a Baxter is not correctly calibrated. The hand annotated with number "1"" in the diagram (i.e the PointCloud) is far away from the actual Baxter's hand (annotated with number "2" in the diagram). This means that the camera's view of Baxter's hand is not at a correct position and hence any hand-pose detected would not be relative either. The interface can be used to adjust the transformation and rotation of the point cloud view so that both the point cloud and the robot model intersect (see Figure 3.8b). Once they are interested it means that the calibration is done.

For better accuracy, Baxter's arm has been placed on the table on purpose so that we can have a sense of the table height.

To achieve this, the script publishes a new transformation from the camera's position to Baxter's base. The camera position is changed based on the sliders adjusted by the user

in the user interface.

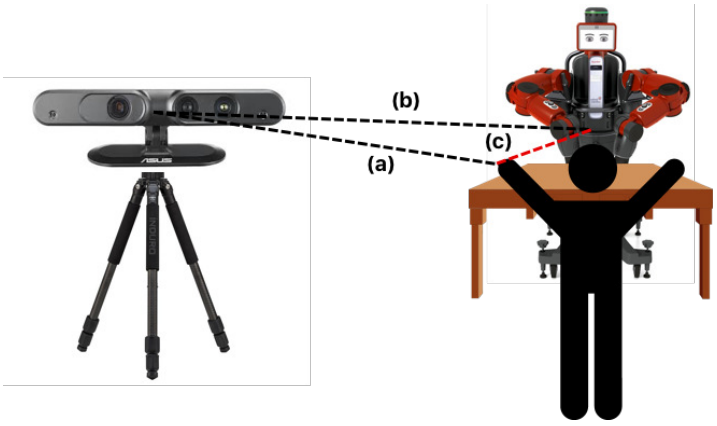


Figure 3.9: Camera Calibration Explained

As can be seen from the Figure 3.9 the dashed line `"a"` is the `tf` identified by the skeleton tracker and is tracking the pose of user's left hand. The script written for the project publishes the `"b"` dashed line. That is, it publishes a `tf` from the camera viewpoint to Baxter's base.

The dashed line `"c"` is not a published `tf`. ROS's tf library can do the transformation and give the relative posture from `"a"` to `"b"`. Since ROS knows the relationship between the camera and Baxter through `"a"` and `"b"` transformations, requesting the transformation from `"a"` relative to `"b"` will result in the transformation of `"c"`.

Auto-save and load functionality: The code has been refactored to become an Object Oriented class. Although the original solution was working as expected, it would be an overhead to have to do this process over and over again.

Given that the camera position (possibly on a tripod) can be placed at the same position over time and that Baxter will also be placed at the same position, the script can save the state of the configuration of the calibration tool so in the future, the operator can load the configuration from last time.

The script will auto-save the values every time the user change one of the sliders. The data are stored on a flat database (text file) at a predefined location with a name provided by the user when initially ran the script. The name will be used later when loading the settings, therefore having a sensible name that describes the location is useful.

When running the script, the `"-l"` flag can be appended at the end to specify that the user would like to load configuration from file. The user will then be prompted to select a file from the list as can be seen from the Figure 3.10:

Advantages: The idea behind this solution is straightforward, and can be easily configured by any user through a friendly user interface. Moreover, this solution does not require any other manual process, like having a QR code attached to the camera to calibrate.

```
[baxter - http://10.0.0.101:11311] rafael@csros03:~/ros_ws$ rosrun baxter_cashier_perception camera_calibration.py -l
Available files:
0. LongRoom.txt
1. RoboticsLab.txt
Enter file number from the list: 0
Start publishing tf...
```

Figure 3.10: The custom script for Image Calibrator during loading from file

Disadvantages: Although auto-save and load functionality is supported, the solution will require manual configuration every time the camera's position change. Moreover, the success of setting up the solution is subject to user's fault. Furthermore, the solution is not very accurate. Since the configuration depends on an image view to intersect with another entity in Rviz makes the accuracy of the relative pose position subject to every configuration and hence accuracy of the solution can not be measured.

Conclusions: The disadvantages of the solution outweigh the advantages and in any other situation an alternative solution would be explored and implemented. However, precise accuracy is not required for the purposes of this project. The simplicity of the solution makes easy to achieve the end-result. Additionally, this solution has been tested and it turns to be accurate enough for the purposes of identifying the hand pose. Exploration and implementation of new ideas had two risks. First and foremost, this problem already added some delays to the schedule and implementing an alternative solution would cause more harm than good. Secondly, the other solutions identified would also have some common drawbacks and the end result probably would not be that much better to worth the time to implement it.

3.2.3 Possible alternative solution identified

Although the solution identified is working as expected it has its drawbacks. When the problem came up, there was two possible solutions, the one already discussed and a more complicated and automated one.

The second more sophisticated solution had to do with a combination of pose estimation using QR or AR codes as well as a help camera.

How the end result is achieved:

As you can see from the Figure 3.11, the idea is that Baxter's head camera is used as a help camera to perform the automatic calibration.

In specific, a QR or AR marker is attached to the RGB-D camera as shown in the figure, which is used by Baxter's head camera to track its position, hopefully giving the actual RGB-D sensor position as well.

The upper dashed line in the figure shows the QR/AR detection where the lower dashed line shows the `tf` published from camera to Baxter's base.

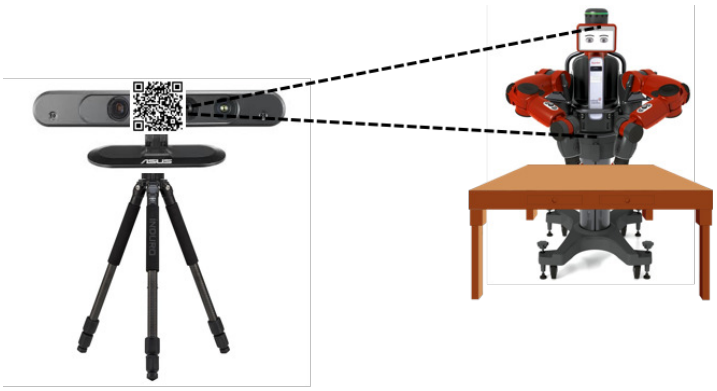


Figure 3.11: Alternative calibration solution using QR Code and Baxter's Camera

There are three hypothesis that needs to be true:

1. The QR/AR Code is clearly visible to Baxter's head camera
2. The QR/AR marker is placed at the correct position and rotation as the camera is looking the world.
3. The camera has a full view of the user so that it can perform its task, which is to track the human skeleton.

The above three hypothesis are mutually exclusive. In specific, the QR code to be in the correct position as the camera does see the world it means it has to be placed exactly above the two image sensors which are looking the user. This means that Baxter will not be able to clearly see the AR code since the AR code will be towards to the user.

Of course, there is a possible solution to this limitation. The marker can be misplaced on the camera and it can be placed at the back, looking to Baxter and then fix this position mismatch using a static transformation.

Advantages: This solution can be considered much more autonomous which is good in an environment like robotics. Being able to have most of the functionalities autonomous and not depend on users configuring the environment is always a plus. The solution requires just a QR/AR code attached to the camera, and in the scenario where the marker is placed at the back of the camera (i.e it does not block camera's view), it can be attached once and will not require to be configured every time, even though the position of the camera changes.

Disadvantages: On the negative side, this solution will require to find the correct fix for both positioning and rotation of the pose. This will be very challenging to find. Not only this, if the QR code position ever changes, it will require new adjustments.

Conclusions Given the time spent on this issue, the complexity, as well as the uncertainty if it will work as expected, led to choose the solution described in Section 3.2.2 instead,

which was already tested by Muhannad in the past.

3.3 Money Recognition

Money recognition is also an important component of the project. To achieve a satisfying end result three solutions have been explored.

3.3.1 Money recognition using colour recognition

Initially, the plan was to use colour recognition to detect different kinds of banknotes.

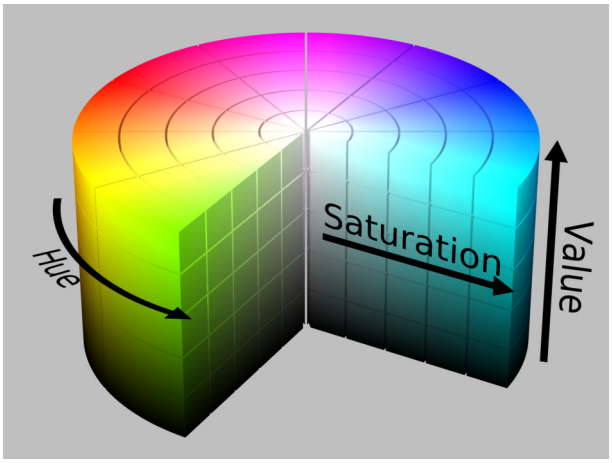


Figure 3.12: HSV cylinder[4]

Hue Saturation Value (HSV) is cylindrical-coordinate colour system. The colour in this system is defined as three values: Hue (H), Saturation (S) and Value (V) [20]. The Hue value represents the actual colour like green, blue, red etc where Saturation and Value represents the colour purity and brightness respectively [20]. HSV was created as a deformation of the RGB system [20]. HSV can therefore be expressed in much more natural way than RGB system. For this reason the colour recognition tested was based on HSV instead of RGB.

The following piece of code, using Python's OpenCV library, takes an image and converts it into HSV-based colour system and tries to detect the colour of the banknote.

Listing 3.1: Colour detection using mask and OpenCV [5]

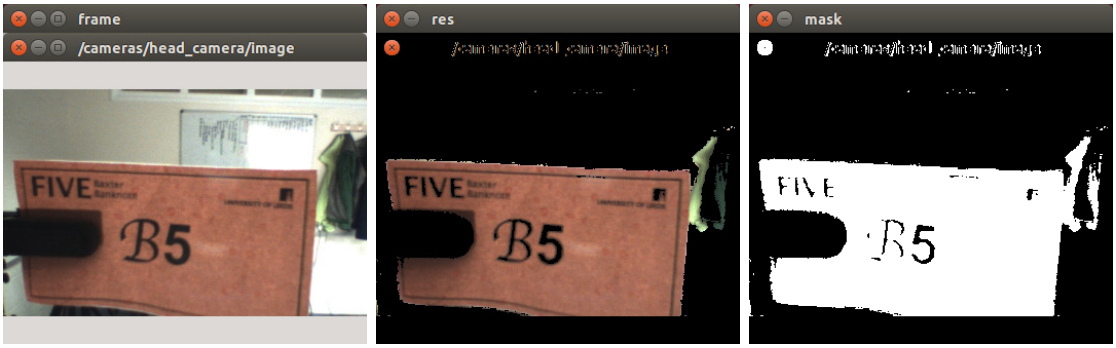
```
1 hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
2
3 lower_range_orange = np.array([1, 100, 100])
```

[4] Picture reference: https://en.wikipedia.org/wiki/HSL_and_HSV#/media/File:HSV_color_solid_cylinder_alpha_lowgamma.png

```
4 upper_range_orange = np.array([101, 255, 255])
5
6 mask_orange = cv2.inRange(hsv, lower_range_orange,
    ↪ upper_range_orange)
```

.

The orange colour is used here to detect the five banknote which has an orange colour. The upper and lower range of the colour is defined in lines 3 and 4. Using the built-in function `inRange`, OpenCV detects the colour defined in the ranges and mask it in the image.



(a) Original Image (b) Masking result (c) End result

Figure 3.13: Algorithm output

As you can see from the Figure 3.13, the algorithm correctly mask the banknote colour and produces an image with only the banknote detected plus some noise from the background.

This is just the representation of the colour detection, how the algorithm concluded to the decision if the banknote in the image is the five or one is straightforward.

The output image was a binary image of zeroes and ones, where ones were the detected colour and zero was any other colour.

A mask for each colour (assuming two banknotes for the project with two different colours) were calculated for each picture, the algorithm simply counted the number of ones in each picture. The mask with the higher counter determines which banknote is presented. If for example the orange mask has more "1"s than the blue mask, that means that an orange banknotes was given (£5). A maximum number of "1"s was also conducted to ensure that the algorithm will also find out cases were the banknote given was invalid or if a banknote was not given at all.

Advantages: The advantage of this solution is that it can be used with any banknotes and is not limited to specific design patterns like the other solutions.

[5] The solution was based Henry Dang's code: `https://henrydangprg.com/2016/06/26/color-detection-in-python-with-opencv/`

Disadvantages: Although in theory such solution sounds trivial, in practice it was not reliable enough to perform the money recognition task. Mainly the light conditions affect the accuracy of the solution. When the solution was tested on the simulator it was accurate as shown in the pictures for example, but when the environment changed, or when the light of the room in the evening reduced the solution started to become less accurate. Not only this, since the background will not be static because the user will be presented in the background, this external factor can have a negative impact on the accuracy of the solution. For example, what would happen if the user wears an orange t-shirt while he has provided a blue banknote (1B)? Probably, the algorithm will detect the orange t-shirt and treat it as an orange banknote (5B) and may give "false positive" outcome. In addition, adding more banknotes to the project like a 20 and 50 banknote will require to find new colours that are not similar and also will eliminate the accuracy of the other existing banknotes since they use a high-range of the available HSV colours. Finally, is worth mentioning that multiple different colours for banknote have been used with high contrast between the two, but the end-result was still not satisfactory.

Conclusions: The main reason why this solution was not used at the end is because external factors like light conditions and background noise affects, in most of the cases, the accuracy of the result. While the light conditions can be fixed by having artificial lighting, the background can not be controlled since the arbitrary customer will be presented in the background and customer's t-shirt may conflict with the banknote colours. Last but not least, a more advanced solution based on colour recognition can be implemented of course (with models), but it will require more time and effort to achieve a satisfactory result.

3.3.2 Money recognition using template matching

Another possible solution for money recognition identified was to use some template (for example the number of the banknotes) and attempt to match the template with the given image. If the template match the image is means that the banknote presented in the scene is the one with the template. Two templates were generated one for each banknote.

A number of tests have been done to evaluate the performance of the solution. For each testing below, image from Baxter's head camera holding a project's banknote was used (to reproduce a real case scenario). The template used in all tests was a screenshot of the centre number of the banknote.

Source Code and Algorithm: In Listing 3.2 the algorithm used for template matching is presented. This simple but straightforward algorithm takes an image and template and using some built-in algorithms, tries to match the template with the image.

Listing 3.2: Template Matching Algorithm [6]

```
1  img = cv2.imread('five.png',0)
2  template = cv2.imread('one_template.png',0)
3  w, h = template.shape[::-1]
4
5  # Apply template Matching
6  res = cv2.matchTemplate(img, template, cv2.TM_CCOEFF)
7  min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
8  bottom_right = (top_left[0] + w, top_left[1] + h)
9
10 cv2.rectangle(img, max_loc, bottom_right, 255, 2)
```

Using OpenCV both the image to be tested and the template to be matched are loaded. On line six of the listing, the actual pattern matching takes action using OpenCV's `matchTemplate` method. The third parameter of the `matchTemplate` method is the template matching algorithm to be used. OpenCV provides a variety of different algorithms to do pattern matching.

Finally, on line 10, a rectangle is drawn on the image to demonstrate the successful detection of the template. If template matching failed to match the image, no rectangle will appear in the image.

Testing and Quality Checks: The first test was to test a banknote against its corresponding template. The expected outcome is that the algorithms identify correctly the position of the template in the picture, which is the centred number (1 or 5).

(a) Correct - Templated detected

(b) False Negative - Template not detected

(c) Template provided

Figure 3.14: Visualisation of the process

Table 3.1 shows each algorithm tested and the outcome (Pass or Fail). Note that the second and third columns of the table show testing on "1" Banknote (1B) and "5" Banknote

[6] Source code reference: `http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html`

(5B) testing respectively.

Table 3.1: Quality check - Correct image

Algorithm	**Pass/Fail (1B)**	**Pass/Fail (5B)**
TM_CCOEFF	Pass	Pass
TM_CCOEFF_NORMED	Pass	Pass
TM_CCORR	Fail	Fail
TM_CCORR_NORMED	Pass	Pass
TM_SQDIFF	Pass	Pass
TM_SQDIFF_NORMED	Pass	Pass

The second quality test done was to try a banknote against a wrong template. The expected outcome is that the algorithms would not detect anything in the picture since the picture does not have the templated provided.

(a) Correct - Nothing detected

(b) False Positive - Template wrongly detected

(c) Template provided

Figure 3.15: Visualisation of the process

Table 3.2 shows each algorithm tested and the outcome (Pass or Fail) again.

Table 3.2: Quality check - Wrong image

Algorithm	**Pass/Fail (1B)**	**Pass/Fail (5B)**
TM_CCOEFF	Fail	Fail
TM_CCOEFF_NORMED	Fail	Fail
TM_CCORR	Pass	Pass
TM_CCORR_NORMED	Fail	Fail
TM_SQDIFF	Fail	Fail
TM_SQDIFF_NORMED	Fail	Fail

Advantages: The main advantage of this solution is that is not limited to a made-up banknote since the template can be defined by the programmer, any banknote (even real one) can be used and it much easier to make Baxter recognise multiple different banknotes using this solution than the colour recognition.

Disadvantages: The solution, as can be seen from the testing tables, is very inaccurate, and is very sensitive to background noise. This gives false negatives and false positives in some cases.

Conclusions: Unfortunately no algorithm was reliable enough and hence this approach could not be used for money recognition. Mainly the background was too noisy and there was no way to bring Baxter's hand closer to the head camera to eliminate the background noise. Finally, the end-result had in some cases false positives which are very critical in this use case (when the user gives no money and the algorithm wrongly detects it).

3.3.3 Money recognition using AR code recognition

The final solution tested was AR Code recognition since the previous attempts failed and the outcome was not well enough. The concept of this solution is to attach AR codes on the banknotes and use an AR Code recognition library to detect the AR code.

The AR codes attached on the banknotes represents numbers. The number that they represent is the amount the banknote worth, so identifying the number of the AR code, would give the banknote number as well.

(a) A five Baxter banknote with AR codes representing the number "5"

(b) A one Baxter banknote with AR codes representing the number "1"

Figure 3.16: The two available banknotes of the project

In Figure 3.16a, as can be seen, two identical AR codes were attached. Since Baxter probably will cover one side when grasping the banknote, the other side will be visible to the AR recogniser. In the project's repository, editable templates are provided. The banknotes are highly configurable (colour changing, attach new AR code etc) so the project can use more than two banknotes if needed. The banknotes are printed two-sided.

How the end-result is implemented: The complete implementation depends on the `ar_track_alvar`[7] library.

The generation of the AR codes was done using the library's node `createMarker`. The marker represents a single digit number. In the case of the "one" Banknote the number

[7] http://wiki.ros.org/ar_track_alvar

"1" and in the case of the "five" banknote the number "5".

The `ar_track_alvar` library by default detects any AR code in the image view. In the case the AR code represents number, it publishes a `tf` with the name `ar_marker_` followed by the number (e.g `ar_marker_1` for one).

The script of the project, when money recognition is requested, will try to look for both `ar_marker_1` and `ar_marker_5` transformations, once it find one will return that value back, performing actually the money recognition task.

Listing 3.3: Banknote recognition

```
1  def detect(self, request):
2    timeout_start = time.time()
3    timeout = 5   # [seconds]
4
5    while time.time() < timeout_start + timeout:
6        # Try to detect either the 5 or the 1 banknote.
7        if self.try_to_detect(5) is not None:
8            return RecogniseBanknoteResponse(5)
9        elif self.try_to_detect(1) is not None:
10           return RecogniseBanknoteResponse(1)
11
12       self._RATE.sleep()
13
14   # If time over and nothing returned, nothing detected and return
       ↪ -1
15   return RecogniseBanknoteResponse(-1)
```

Therefore, this solution depends on the `ar_track_alvar` library that runs in the background and detects AR codes. When requested it will try to do tf-lookup for `ar_marker_1` or `ar_marker_5` (in the Listing 3.3 the tf-lookup is done using the method `try_to_detect(n)`). If the marker is not presented in the scene, then the tf-lookup will fail and will not return anything. If however, a marker is presented, it will return the value of the marker and hence the value of the banknote.

The algorithm has a five second window for the detection, but if the `ar_track_alvar` detect the marker faster then the algorithm will return the value faster. If however, nothing detected within the five seconds, the algorithm will return -1 and hence the indication that nothing has been detected.

Advantages: This solution is the most accurate over the previous two solutions. To put it in context, this solution was tested more than 15 times, during the general testing of other parts, in every case it achieve 100% accuracy! The testing was done in different times

of the day, with several corner cases (giving no money, upside-down etc). The algorithm is working perfectly fine in all the cases that can happen in this context: valid banknote presented, invalid banknote presented, no banknote presented, banknote presented but upside-down, in every case the algorithm respond accordingly. Last but not least, this solution will work with an unlimited number of different banknotes, given that each uses a different AR code number making the project much more flexible than the other solutions identified.

Disadvantages: The only disadvantage of the solution is that it requires the AR code on the banknote which does not allow space to use real banknote with the project, and also requires the AR codes to be printed on the banknotes which make them not too realistic.

Conclusions: Since the objective of the project is to implement human-robot interaction with banknotes as the mean of the interaction, and the motivation is not to use real banknotes, this solution is by far the most appropriate since it is very accurate and it does not have any false positives. The accuracy during the testing was impressive, and the end-result satisfactory.

3.3.4 Final thoughts on money recognition

Given the time left for the project and the requirements of a very stable solution that will always work and will not depend on external factors like the background noise or the light conditions, the best solution was AR mark detection.

Although a custom algorithm that would detect project's banknotes would be more attractive than a third-party library that does the recognition it also has it disadvantages. Any custom solution would require dramatically more time to implement and test, and possibly would not be that accurate as a library written some years ago by many individuals and tested by tens of individuals.

3.4 Feedback output

Feedback output has been implemented to make the user aware of some events during the interaction.

In addition to the above indications, another two important screens are the "amount due" and "change due" screens.

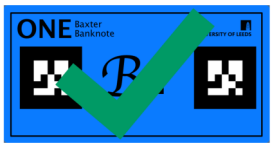
In Figure 3.18 the images are generated dynamically using Python and OpenCV library to draw images and text on a template to achieve the end result of the amount due and change due screens. The amount due screen shows the amount that the user owns to

(a) Focusing eyes

(b) Focusing eyes (right)

(c) Focusing eyes (left)

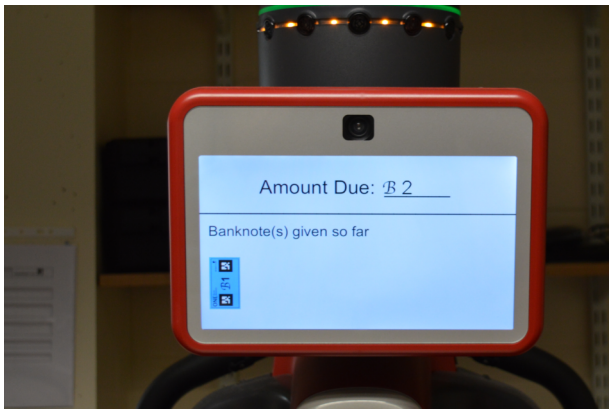
(d) Banknote recognition success (1)



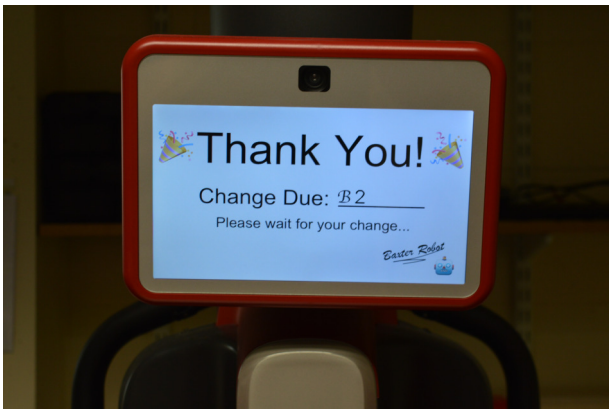
(e) Banknote recognition success (2)

(f) Banknote recognition fail

Figure 3.17: Baxter Screen Feedback Output (Interactive eyes showing movement and successful/fail banknote recognition)



(a) Amount Due



(b) Change due

Figure 3.18: Change and Amount due screens

Baxter and underneath the amount due are the banknotes given by the user so far in order. The "change due" screen, on the other hand, is indicating to the user the change should expect from Baxter.

The importance of feedback output in this context: When Baxter is performing a time-consuming task like when recognising banknote usually it takes several seconds to complete. Baxter uses "animated" eyes (animated is achieved by showing a sequence of images the one after the other where the eyes look left then right and so on) when performs a time-consuming task to indicate to the user that the robot is **not** stopped.

Chapter 4

Manipulation Development

4.1 MoveIt!

As described in Section 2.7.2, "MoveIt!" is a third party library that provides planning and collision-aware planners. "MoveIt!" has been used in this project for manipulation and obstacle avoidance.

4.1.1 Manipulation using MoveIt!

The script `moveit_controller.py`[1] was written to wrap "MoveIt" into a friendly interface so it can be used throughout the project without having to repeat the code.

This script contains several classes but the main one is the `MoveItPlanner` which contains the following help methods:

1. `_create_scene`. This method is the one that will create and publish the obstacles to MoveIt's world. This method runs on the construction of the class, so the obstacle publishing is done automatically.
2. `is_pose_within_reachable_area`. Checks whether the given pose is within the Robot's reachable area or not as described in Section 3.1.2.
3. `move_hand_to_head_camera`. This method will move the active hand (active hand is the hand that was used the last time to perform a task) to Baxter's head camera. This is especially useful for the money recognition.
4. `move_to_position`. Given a pose and arm, this method will move Baxter's given arm to the pose.
5. `leave_banknote_on_the_table`. As the name implies, this method will move Baxter's active hand to a position where the grasped banknote can be left on the table.
6. `set_neutral_position_of_limb`. This method will move the active hand to its neutral position.
7. `get_end_effector_current_pose`. Using forward kinematics, this method will return the pose of the current arm state.

[1] `https://github.com/papallas/baxter_cashier/blob/master/scripts/baxter_cashier_manipulation/src/moveit_controller.py`

8. `open_gripper` and `close_gripper`. As the names imply these methods perform an open/close operation to Baxter's end-effector.

The above methods provide all the functionalities needed for the project in terms of manipulation. Some generic methods like the one that moves a hand to arbitrary pose and some specific methods like the one that leaves the banknote on the table. Since the script publishes the obstacles to MoveIt!, MoveIt! planner will automatically consider the obstacles during the planning.

This class is used in the main script and is responsible for performing the manipulation tasks of the project.

4.1.2 Obstacle Avoidance

During the general testing of the project, in some cases, Baxter was very close to hit the table and also the side wall. Since Baxter's default planner was not aware of those obstacles there were concerns that the robot might damage itself during the manipulation.

A possible solution was to hard-code some values that would not allow Baxter's arms to go underneath the table for example. However, this not only makes the visualisation of the obstacles harder but also if Baxter ever change environment then the demonstrator will have problems.

Baxter's default planner does not provide any mechanism or API to define obstacles and hence make the planner collision-aware. So the use of the third-party library "MoveIt!" is used. This library was discussed in the Section 2.7.2.

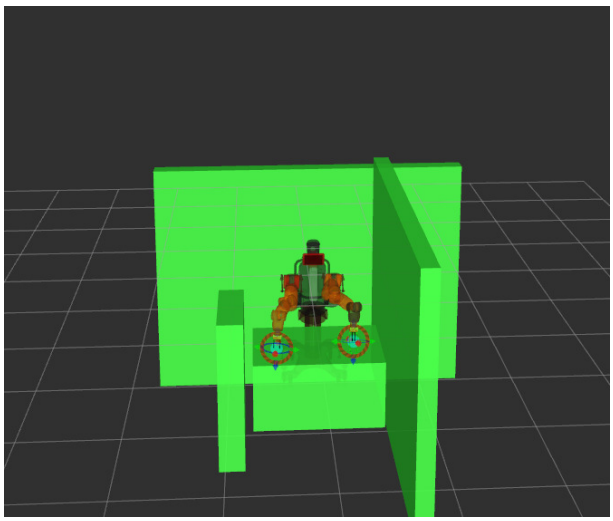


Figure 4.1: Rviz: Baxter surrounded by obstacles

As can be seen from Figure 4.1 Baxter is surrounded by four obstacles: a side-wall, a back-wall, a table and a tripod.

The obstacles are manually defined to reflect the real environment Baxter is currently placed.

Flexibility of changing obstacles

As already mentioned, the reason for choosing such sophisticated and complicated solution as MoveIt! is to allow flexibility when changing Baxter's environment. For example, if Baxter is moved to another room, updating the obstacles should be easily possible.

The file `environment_factory.py` located at the manipulation package of the project is a custom written script that does the job of defining obstacles to Baxter's world. This class uses a combination of two Software Engineering Design Patterns: **Template** and **Factory** design patterns.

This file has four main classes:

- **EnvironmentFactory**. Provides the interface to request individual environments.
- **Obstacle**. Describes a single obstacle in the world. The definition includes the name, pose as well as the dimensions of the obstacle.
- **Environment**. Is the template class that discrete classes will inherit from. Defines "required" attributes as well as two methods: `clone` and `get_obstacles`.
- **Discrete class**. Each real-world environment like Robotics Lab, Long Room etc would be defined as a discrete class defining their obstacles.

Listing 4.1: A part of the discrete class that defines Baxter's current environment

```
1 class RoboticsLabEnvironment(Environment):
2     def __init__(self):
3         self._obstacles = []
4         self._create_obstalces()
5 
6     def _create_obstalces(self):
7         side_wall = Obstacle(obstalce_name="side_wall", x=0.6, y=1,
              ↪ z=0, shape_size=(4, 0.2, 3))
8         self._obstacles.append(side_wall)
9         # ... etc
```

The file `moveit_controller.py` is the script that creates and configures MoveIt!. The script also provides an interface to use MoveIt! with the project's main class. In this file, the publishing of obstacles is taking place which in turns makes Baxter obstacle aware.

4.2 Money grasping and change handling

Another important aspect of the project is money grasping from the table and change handling.

4.2.1 Money configuration

A hard problem that this project faced is how to pick a very thin material like a banknote from the table. Unfortunately, Baxter at the School of Computing does not have extra end-effectors that will allow manipulating such thin material.

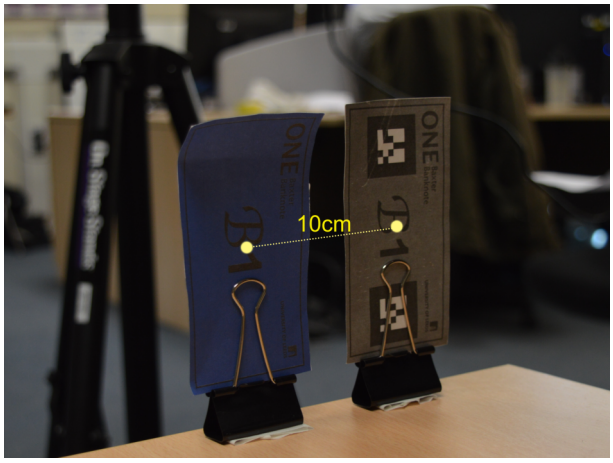


Figure 4.2: Banknote Standing on the table

The solution identified was to use binder clips to vertically hold the banknotes on the table as like in Figure 4.2. The binder clip in this project will refer as banknote stand.

4.2.2 Pose estimation

There are an undefined number of banknotes on the table. Undefined means that Baxter can work with any number of banknotes on the table on demand.

First of all, both sides of the tables assume to have at least one banknote. At the start of the project (when user run the project) is asked to move both Baxter's arms above the left-most banknote of each side of the table and then hit enter. The user will also be asked to enter the number of remaining banknotes on the table for each side.

Baxter will record the pose of the first banknote of each table side using Forward Kinematics to work out the pose of the arm's current pose. Will then automatically work out the pose of the remaining banknotes.

Baxter iteratively will pick the previous banknote pose and append a 10cm to the x value of the pose for the right side of the table. This assumes that all banknotes have an equal spacing between each other, with the centre point being 10cm away from the previous banknote. For the left side of the table however, the banknotes are placed horizontally and hence the 10cm is added to the y value of the pose. The left side was placed horizontally because on this side of the table the sweet bowl for the shopkeeper demo is placed (will be discussed in Section 5.2).

4.2.3 Logic for change handling

The main script of the project handles the logic for change handling.

A class variable called `amount_due` is maintained. This variable can be either positive, zero or negative value.

- **Positive** value, means that the customer owns money to Baxter, and hence Baxter will trigger the "pick" method, to get money from the customer.
- **Negative** value, means that Baxter owns money to the customer, and hence Baxter will trigger the "give" method, to return change to the customer.

The algorithm uses iterative approach until the amount due variable becomes zero. Zero means that both sides are satisfied and hence the interaction is over.

Use case

John is a customer and he orders three sweets from Baxter, one blue one red and one green. The grant total of John's order is 3 Baxter banknotes. The amount due variable is set to +3.

Since the amount due is positive, Baxter waits for John's hand pose. John gives a five banknote to Baxter. Baxter through his banknote recognition module recognises that the given banknote has the value of 5. It then subtracts this value from the amount due, and hence the amount due becomes −2.

Since the amount due is negative, Baxter understands that he owns money to John. Baxter twice picks one, "1" banknote from the table and returns change to John. After giving two "1" banknotes the amount due becomes zero both Baxter and John are happy and the interaction is over.

Chapter 5

Final Solution Architecture & Integration

5.1 Final Solution Architecture

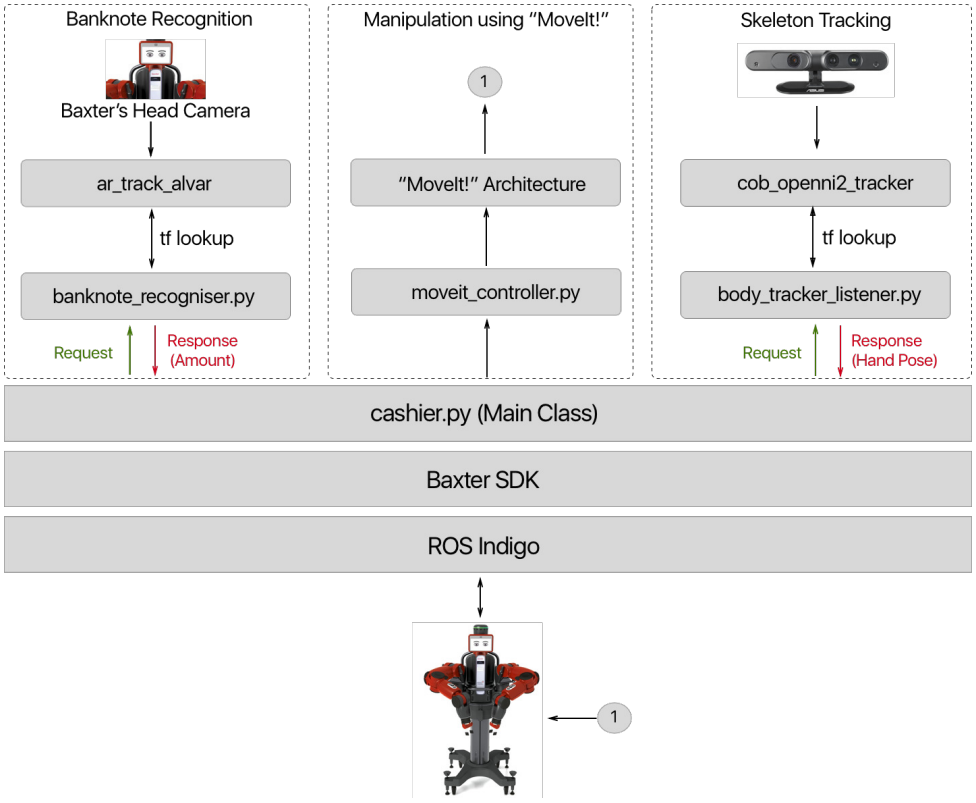


Figure 5.1: Project's Architecture

The final solution has four main components.

1. Main script that handles the logic and connects different components under a unified script.
2. Banknote Recognition that handles the money recognition.
3. Manipulation using MoveIt, that does the core manipulation of the project.
4. Skeleton Tracking that tracks hand poses.

These four components sit on top of two middlewares. Baxter's SDK that provides the interface to interact with Baxter and ROS that does the lower-level work. MoveIt could also be considered as middleware although since was wrapped into a custom class which is callable within the main script, it is better (at least in terms of visualisation) to presented as a component.

As can be seen from Figure 5.1, Banknote Recognition, Manipulation and Skeleton Tracking are surrounded by dotted borders. The main script (`cashier.py`) knows nothing about the architecture of these three components, it only knows about the custom scripts `banknote_recogniser.py`, `moveit_controller.py` and `body_tracker_listener.py`. These scripts were written for this project to hide the implementation details of the corresponding architectures and to expose a clean and readable interface to be used in the main script. This also separates responsibilities and makes easier to read the code.

5.1.1 Money recognition

As described in Section 3.3, money recognition after many experiments conclude to use the AR recognition described in detail in Section 3.3.3. As can be seen from Figure 5.1 the custom class `banknote_recogniser.py` was implemented as a service that on request returns back the amount of the recognised banknote using `tf-lookup` to the `ar_track_alvar` library.

5.1.2 Manipulation using "MoveIt!"

As already discussed in Section 4.1, "MoveIt!" framework is used as the core manipulation mean of the project, which also provides a collision-aware planner that can avoid obstacles in the scene. Main class instantiate an instance of the class located in the `moveit_controller` script. Then the main script uses all the help methods provided by `moveit_controller` to perform the manipulation tasks of the project.

5.1.3 Skeleton Tracking

Skeleton tracking is the essential task that recognises the hand-poses of the users. This is done using `cob_openni2_tracker` for the reasons described in Section 3.1. Again the custom script `body_tracker_listener.py` was implemented as a service that on request returns back the hand pose.

5.1.4 Project's Overall Logic

In Figure 5.2 is the project overall logic in a sequence of steps to achieve the end-result.

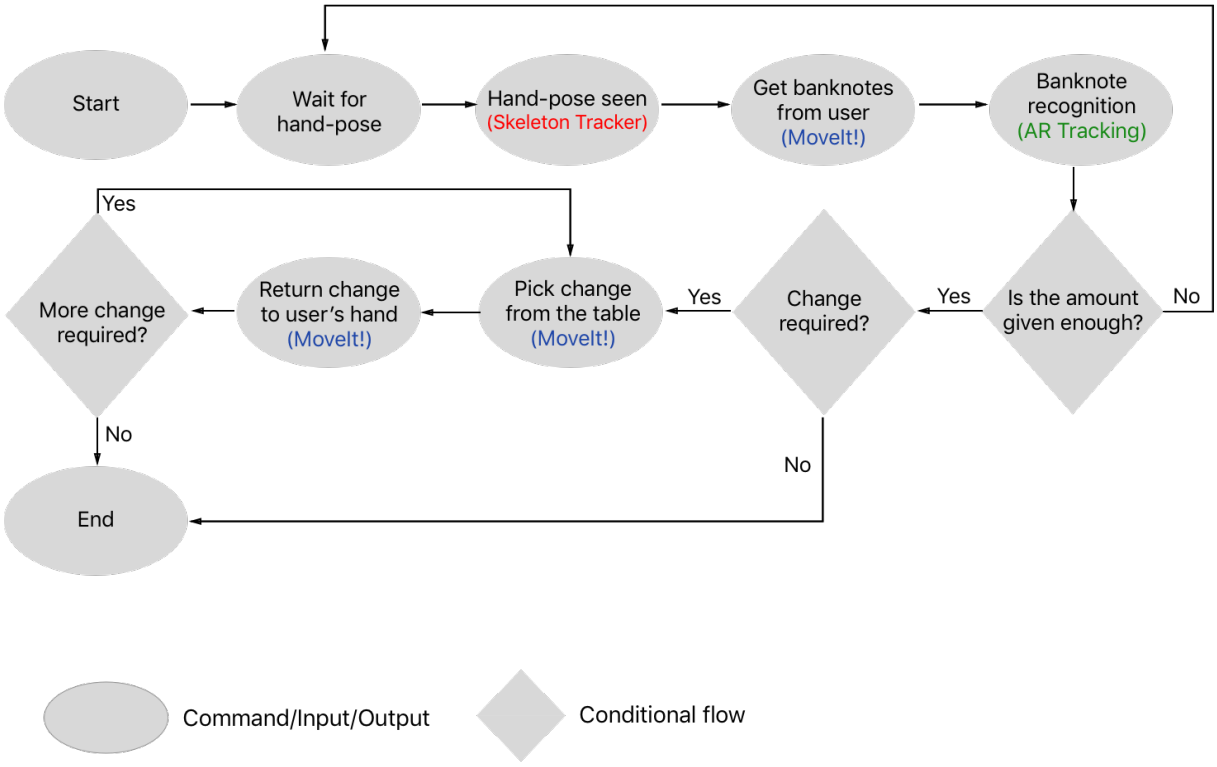


Figure 5.2: Project System Logic

5.2 Integration with last year's project

At the end of the core development of the project, another objective was to merge the last year's project with this project.

5.2.1 About last year's project

Last year a student completed a project on Baxter that received orders from a customer through an Android application. Baxter is able to take a bowl with sweets, detect the different coloured sweets and pick the ordered ones. Baxter then drops those sweets in a customer bag ready for the customer to collect.

This year as already explained, the objective was to develop a human-robot interaction with Baxter as a cashier. The goal of this section is to explore how the integration was done to implement a unified project that takes and prepare the order and handles payment with an arbitrary customer.

5.2.2 How integration was done

There were two ways that the integration could be done.

1. Integrate last year project logic in this year's project.
2. Integrate this year's project logic in last year's project.

The best way identified is that it could be easier and more logical to integrate this year's project in last year's project. Both projects will live in separate packages, however.

In addition, from the beginning when the project has started the need for integration was known and hence the project was designed and implemented in such a way that will make easy to run the project's code within another script. This was achieved by making the project object oriented.

Figure 5.3 shows the overall system logic, including the last year's project. The dotted border is the logic of this project appended on top of the last year's project.

As with a real customer-cashier interaction, the order should be first prepared before the customer proceed with the payment. Therefore, this project logic was placed after the sweets have been placed into the bag and just before the customer leave. The diagram shows exactly where this year's code was placed.

A forked version of last year's project was created so it can be altered with the changes required to do the integration with this project. (How to access this forked repository see appendix C)

In terms of code integration, due to the fact that this project was entirely object oriented, it was a matter of three lines of code to do the integration:

Listing 5.1: Code integration appended in last year's project

```
1 cashier = Cashier() # Object instance of the project.
2 cashier.amount_due = grand_total
3 cashier.interact_with_customer()
```

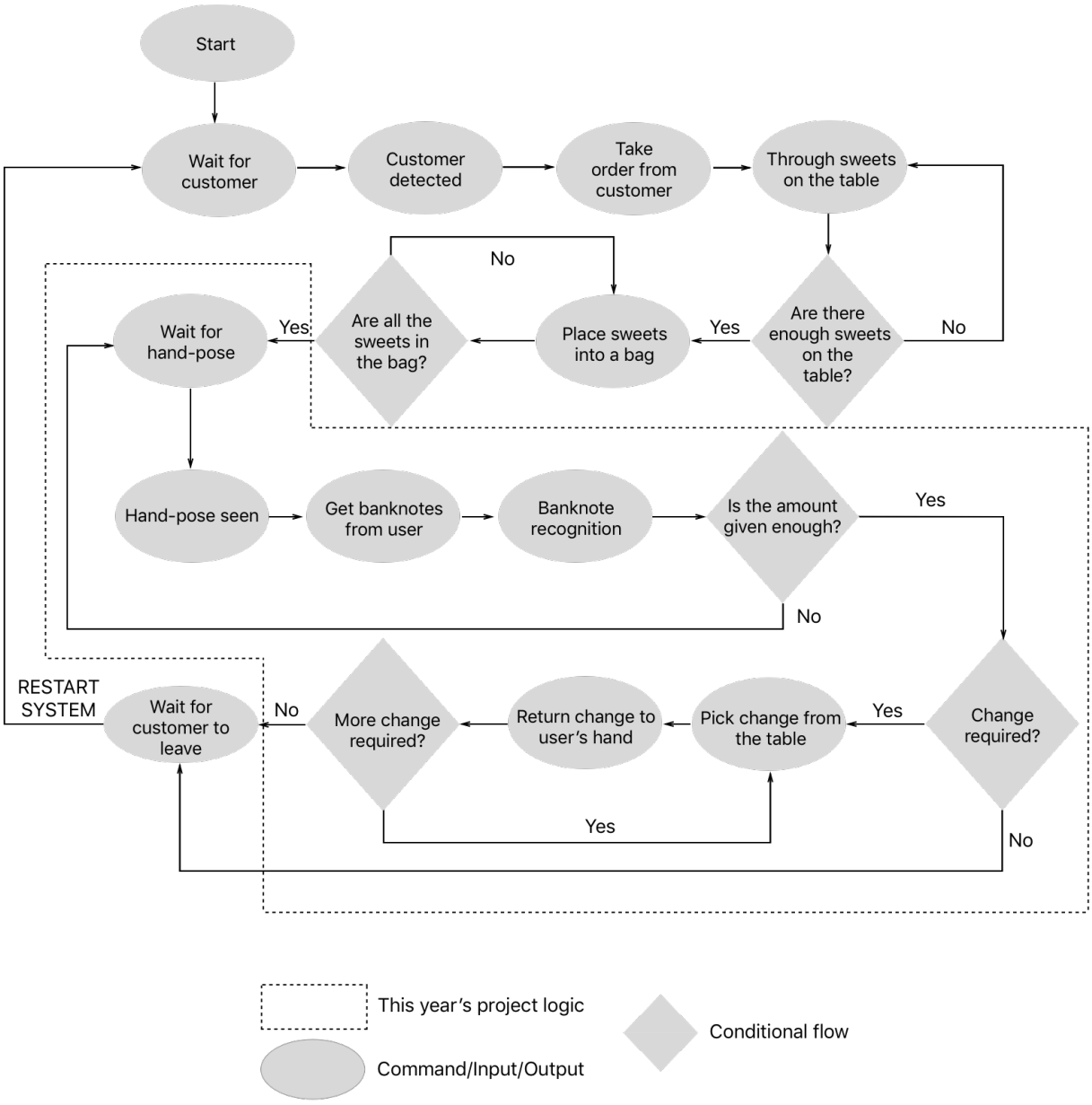


Figure 5.3: Overall System Logic after the integration

Chapter 6

Testing & Evaluation

The testing was split into two phases. First and foremost, ten participants were invited to do the testing and share their thoughts and their feedback.

In addition, Unit Testing also has been made to test individual parts of the project in more detail and also to test some corner cases not covered by the participants.

6.1 Participants Testing

Ten participants have been invited to test the project and also to provide their feedback and their opinions about the project. At the end of the interaction, each participant completed the Evaluation Form (appendix G).

These ten participants have been used in a variety of different test cases. One of the participants had the chance to test the entire project including the last year's project integrated to a final demo and the remaining nine tested just the Cashier part. The nine participants have been used to test cases where they had to give the exact amount (therefore no change to be returned), a case where they had to give more and therefore change to be returned and a variation of the two with different amounts.

For each participant, the interaction was video recorded. The main reason is to explore and analyse each specific interaction in detail, measure the time response of individual modules and use some screenshots of the interaction after the approval of the participants.

Most of the participations succeed and just a few had some problems. The main problems that occurred were:

- The way participants gave the banknote to Baxter's grippers was not too tight and the banknote falls a bit and as a result, the money recognition failed in some cases.
- Some participants were too tall or have raised their hands too high, outside the reachable area.
- Sometimes the participants, as they were not aware of the reachable area and that they have to enter the reachable area when they want to perform the banknote hand-over, they mistakenly trigger the robot's pick algorithm.
- Sometimes the robot was a bit offset from the actual hand-pose (very small offset)

and some participants thought that Baxter will fix this offset and update the positioning to go exactly where their hand was. However this was not the case, there was no such mechanism and as a result, they have missed the chance to hand over the banknote.

Finally, most of the interactions took at most 1 minute and 30 seconds depending on the interaction. Some interactions took twice the time because there was more banknote handover than other interactions.

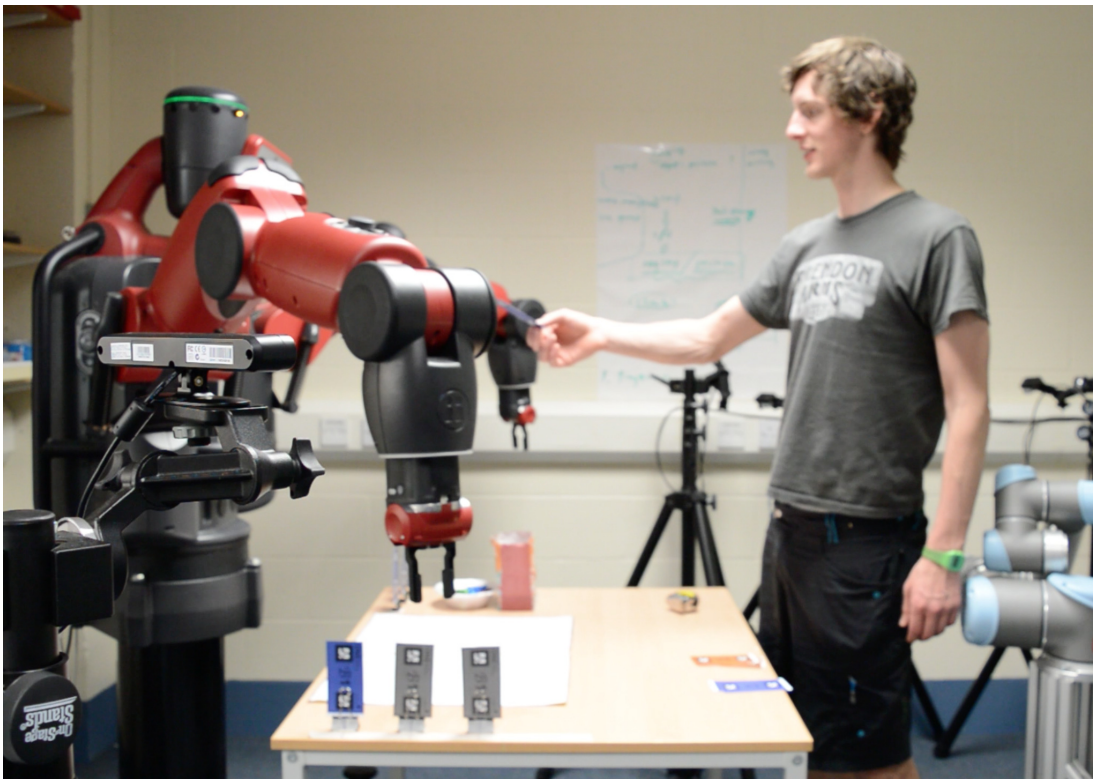
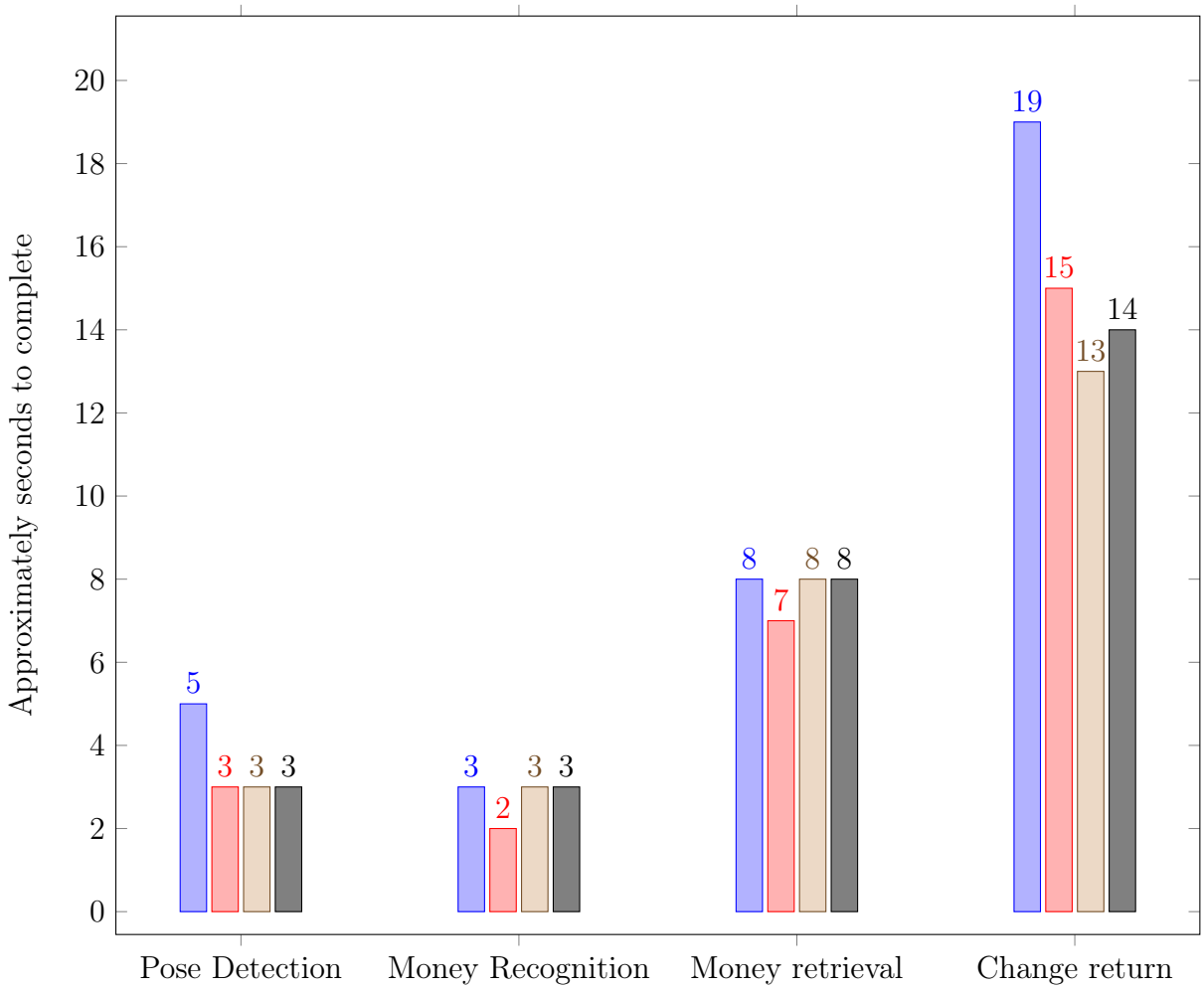


Figure 6.1: Participant too tall and hand-pose out of reachable area

6.2 Timing breakdown



The time breakdown chart above shows the time measurement of each individual module/unit during the interaction of four participants.

On average, Pose Detection took 4 seconds, Money Recognition took 3 seconds, Money retrieval took 8 seconds and Change return took 15 seconds to complete on average. With that said, a normal interaction with the Baxter cashier that involves one time the four activities above takes 30 seconds to complete on average. Usually, most of the interactions will involve twice the tasks above which translates to 1 minute on average.

The following formula will give the running time of the demo on average given some values:

$$\overbrace{p \cdot n1}^{PoseDetectionTime} + \overbrace{m \cdot n1}^{MoneyRecognitionTime} + \overbrace{r \cdot n1}^{MoneyRetrievalTime} + \overbrace{c \cdot n2}^{ChangeReturnTime}$$

where p is the pose detection average time (4 seconds), $n1$ is the number of banknotes given by the customer, m is the money recognition average time (3 seconds), r is the

time to retrieve money from the customer (8 seconds) and finally c and $n2$ is the time to return change to customer and the number of banknote to return respectively.

6.3 Discussion of participants feedback and results

Each participant has been given a participation feedback form to complete. This form was asking the participants several questions regarding the interaction.

Robot's arm positioning

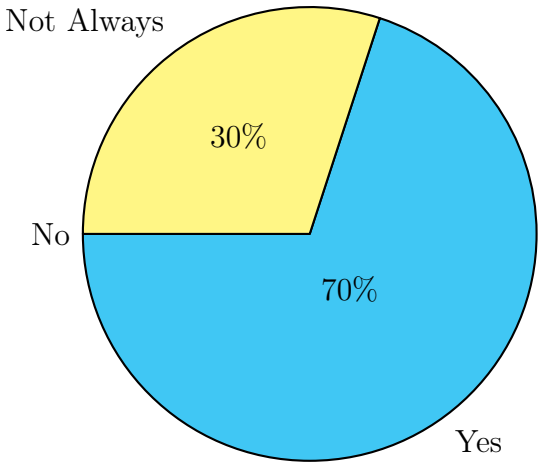
To the question, if the robot's arm positioning was accurate and of their convenience, 70% reported that the positioning was accurate where 30% reported that the positioning was accurate most of the times but not always. None have reported that the positioning was not accurate at all.

The main reason why the 30% reported that the positioning was not always accurate, after watching the interaction video again, is because people moved their hands in the reachable area before they were ready to perform the hand-off activity. The participants have not been introduced to the two-way pose elimination or any specific technology behind the project. When the participants felt that the positioning was not accurate it was when they have triggered the robot without their intent by moving their hands within the reachable area, this is clear because the participants were lost and they have not understood that they had to give the banknote to Baxter at the time. A possible improvement to overcome this problem is to develop voice output that will explain to the user that they have to pay the amount due by raising their hand towards the table when they are ready.

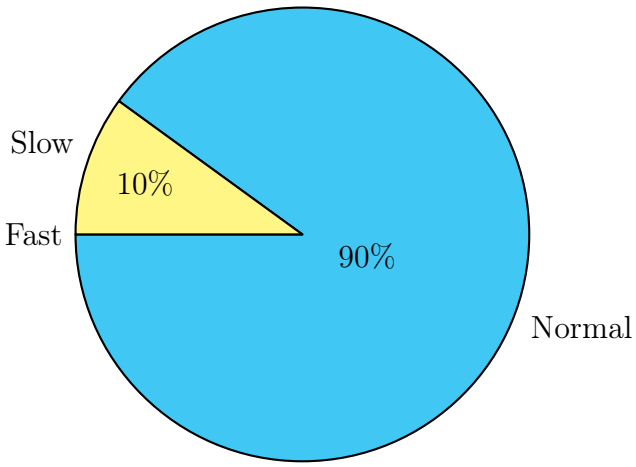
Robot's speed & Interaction quality

In regards to the speed of the robot and the interaction in general, "Normal" speed is the dominant opinion with nine out of ten and just one out of ten reported that the speed was "Slow". None have reported that the robot was fast.

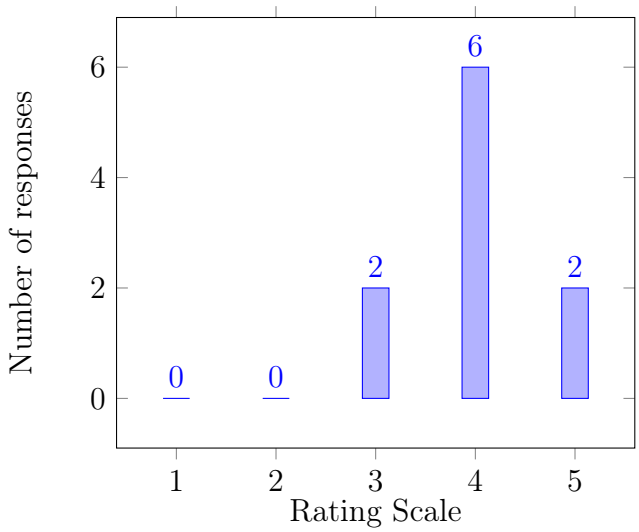
In addition, participants have seen the interaction with positiveness. In specific, eight out of nine participants have reported 4 and 5 (i.e Very Good and Excellent) with just two participants that have found the interaction in the middle. The only comment here is that none have reported that the robot was fast. This is true since the robot takes a while to recognise the hand-off intent and also to complete the interaction. Normal speed is a more desired speed than fast because being fast may cause other problems.



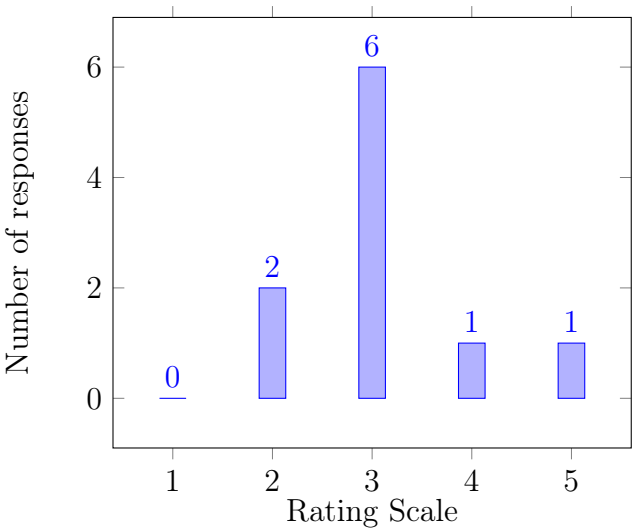
(a) Was the robot's arm positioning accurate and of your convenience when the robot gave/took money from/to you? (No is 0%)



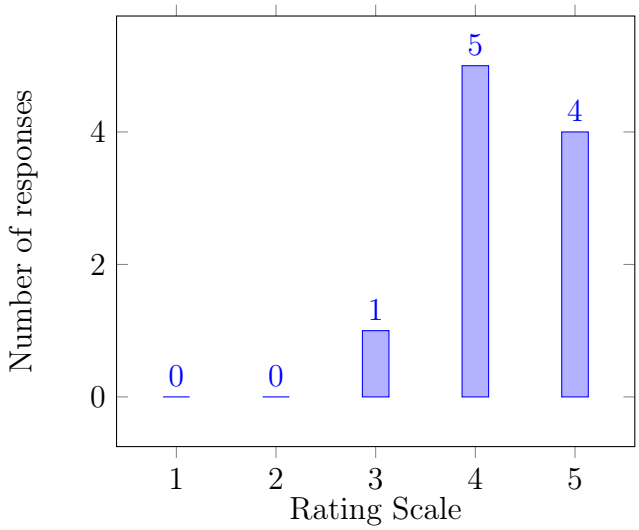
(b) How would you rate the speed/responsiveness of the robot during the interaction? (Fast is 0%)



(c) How good was the interaction? (1 for "Bad" and 5 for "Excellent")



(d) How similar was the robot compared with a real cashier? (1 for "Not Similar" and 5 for "Very Similar")



(e) How clear was the process during the interaction? (1 for "Not clear" and 5 for "Very clear")

Figure 6.2: Participants feedback form results

Similarity & Clarity

Surprisingly, most of the participants have mixed feelings on whether the interaction was similar with a real cashier or not, with six participants reporting "3" on the scale 1 for "Not Similar" and 5 "Very Similar", leaving a neutral impression. Two participants even voted a lower value of "2"". Just two participants have reported "Similar" and "Very Similar" among the ten.

The main problem that most of the participants have been faced is that they did not found natural to extend their hand and wait for the cashier to reach them to pick the banknote. On the other hand, some participants have found this very natural. The ones that have not found the interaction natural they expected to reach the robot instead or at least the robot to make a semi-reach activity to show to the user that the robot is waiting for hand-over activity. This seems like a good observation, however, there is a possibility that both the user and the robot will start moving towards to each other and may cause confusion. That is, some users may understand that they have to reach the robot to give the banknote where in reality the robot will reach them.

Finally, to the question whether the process was clear to the participant, all of the participants have reported "Clear" and "Very Clear" with just one reporting neutral.

Conclusions and feedback ideas

With all that said, the testing was good but the project have the potential for improvement. The bigger problems that the participants have found is that the cashier did not offered a natural interaction. A participant has reported that he was expecting to give the banknote to the robot's arm instead of waiting for the robot to detect his hand and approach him.

Some participants have left comments and feedback for improvements. Four participants have expressed their opinion that although the screen output was clear Baxter could also use voice output to let the users know that they have to take action. Another participant has reported that it would be good if Baxter's pick algorithm could detect when the user has actually given the banknote instead of having to go through the banknote recognition to detect that the customer has not given the banknote.

6.4 Unit Testing

In the previous two cases, the overall systems were tested on how successful components worked together. In this section, the testing is zoomed into individual components like

Money Recognition, Hand-Pose detection etc. Not to be confused with Software Engineering "Unit Testing" where automated tests are written. Here the term "Unit" refers to individual components as opposed to the overall integrated system.

6.4.1 Skeleton tracking and Hand-pose detection

Table 6.1: Testing results Pass/Fail

Test Case	**Outcome**
Hand pose detection is accurate for user's left hand	Pass
Hand pose detection is accurate for user's right hand	Pass
Hand pose detection skip pose outside the reachable area	Pass
Hand pose detection skip pose inside the reachable area but moving fast and not in the hand-pose state	Fail in some cases

The skeleton tracker passed all the tests instead of the last one with the reachable area and moving the hands fast in that area. It turns out that the skeleton tracker reads very fast and even if moving the hands very fast it will trigger the robot if the hand is in the reachable area.

This should be an easy fix if the number of consecutive frames increased by 3 more consecutive frames, however, it will also cause more delays in other cases.

6.4.2 Money recognition

Table 6.2: Testing results

Test Case	**Outcome**
One Banknote was recognised successfully	Pass
One Banknote was recognised successfully (upside-down)	Pass
Five Banknote was recognised successfully	Pass
Five Banknote was recognised successfully (upside-down)	Pass
Fake banknote was not recognised and error is shown	Pass
No banknote provided, error is shown to screen	Pass

The money recognition succeeds in all of the testings however one of the weak points identified, is that when the banknote is two-side printed on an A4 paper, the recognise module sometimes fail to recognise the banknote. New banknotes have been printed with a thicker material during this testing and it seems to fix the problem, however, sometimes during the participants testing there were some false negatives.

Overall, the money recognition works as expected, it never showed a false positive which is the most important point.

6.4.3 Manipulation using MoveIt

Table 6.3: Testing results Pass/Fail

Test Case	**Outcome**
MoveIt is aware of obstacles	Pass
Estimation of remaining banknote on the table succeed	Pass
Change pick-up succeed	Pass
Baxter is aware of missing banknotes from the table	Pass
Both arms can handle change and money pick up	Pass

Although MoveIt! does some great work to the manipulation part of the project some weak points have been identified here as well. When the full environment (including the last year's project objects) was prepared (bowl with sweets, bag for the sweets, banknotes on the table etc) the free space of the table was dramatically reduced which in some cases had Baxter hitting some of those objects, especially when leaving the banknotes on the table.

This could be fixed if there was a way to detect the objects and add them as obstacles or if those objects were in static position and have been defined as obstacles from the beginning.

Chapter 7

Methodology & Project Management

In this chapter the methodology and project management of the project will be discussed, including how the project was divided into different phases, general information regarding the approach taken to keep the software version controlled as well as other project management techniques.

Furthermore, in appendix D is the Gantt chart with the project's progress, that shows the week breakdown and the time spent on different tasks.

7.1 Methodology

The project followed a phase-driven development more close to Scrum methodology, although since there was only one person responsible for the development, the approach can not be described as such.

The project was split into three phases each of which completed a major milestone. First and foremost, phase one brought a basic solution that had major libraries integrated like the camera driver and skeleton tracker as well as implemented features like hand-pose detection, banknote retrieval from user's hand and basic recognition of the banknote.

The second phase brought some critical and important features to the project like collision-aware planner, more advanced and accurate money recognition as well as change handling.

Last but not least, phase three dealt with integration and testing of the project as well as some minor improvements.

7.2 GitHub Repository

The project is hosted on GitHub as a public repository[1]. GitHub provides several project management features that have been used to keep the project under control.

[1] `https://github.com/papallas/baxter_cashier`

7.2.1 Open Source License

The GitHub repository as already mentioned is a public repository which means that anyone with or without permissions can access the source code, wiki and anything that is stored in the repository. The intent of having the repository open to the public is for mainly two reasons. First and foremost, if in the future a student of the University of Leeds will work on a similar project that will have the potential to integrate with this project, it will make it easier for the potential student to access the code and do the integration. Finally, since Baxter is a robot that is all around the world, other universities and researchers can use the project's codebase to improve it.

The repository, as well as all of the project's source code, is licensed under "GNU General Public License v3.0"[2].

7.2.2 Version Control

Branch-per-feature philosophy has been used to separate the major feature implementation and different experiments. This made the entire development of the project much safer since such philosophy promotes experiment and change attitude. The project made use of 11 branches in total[3].

It is worth mentioning the case where the project needed to become obstacle aware. For this major milestone, there was the needed of experiments with the possibility of breaking some existing functionalities. The need of an independent branch was therefore required. Two branches were introduced for this reason one called `phase2-obstacle-avoidance`[4] and one called `phase2-obstacle-avoidance-using-moveit`[5]. The latter was specifically about "MoveIt!" library experiments were the former was in general about the general feature. "MoveIt!" worked as expected and hence there was no need for further branches for this feature.

Other good practices like frequent commits have also been used. To put in in context there are more than 200 commits (including merge commits).

7.2.3 Wiki Pages

A major goal was to maintain a very detailed wiki page that will explain in detail how to install dependencies, understand of fundamental tasks to have the project running and

[2] `https://github.com/papallas/baxter_cashier/blob/master/LICENSE.md`
[3] `https://github.com/papallas/baxter_cashier/branches`
[4] `https://github.com/papallas/baxter_cashier/tree/phase2-obstacle-avoidance`
[5] `https://github.com/papallas/baxter_cashier/tree/phase2-obstacle-avoidance-using-moveit`

running the project itself.

7.2.4 GitHub Pages

Although optional, a welcome website[6] for the project has been created. The motivation behind the website is to show in brief details about the project (including pictures, videos and text).

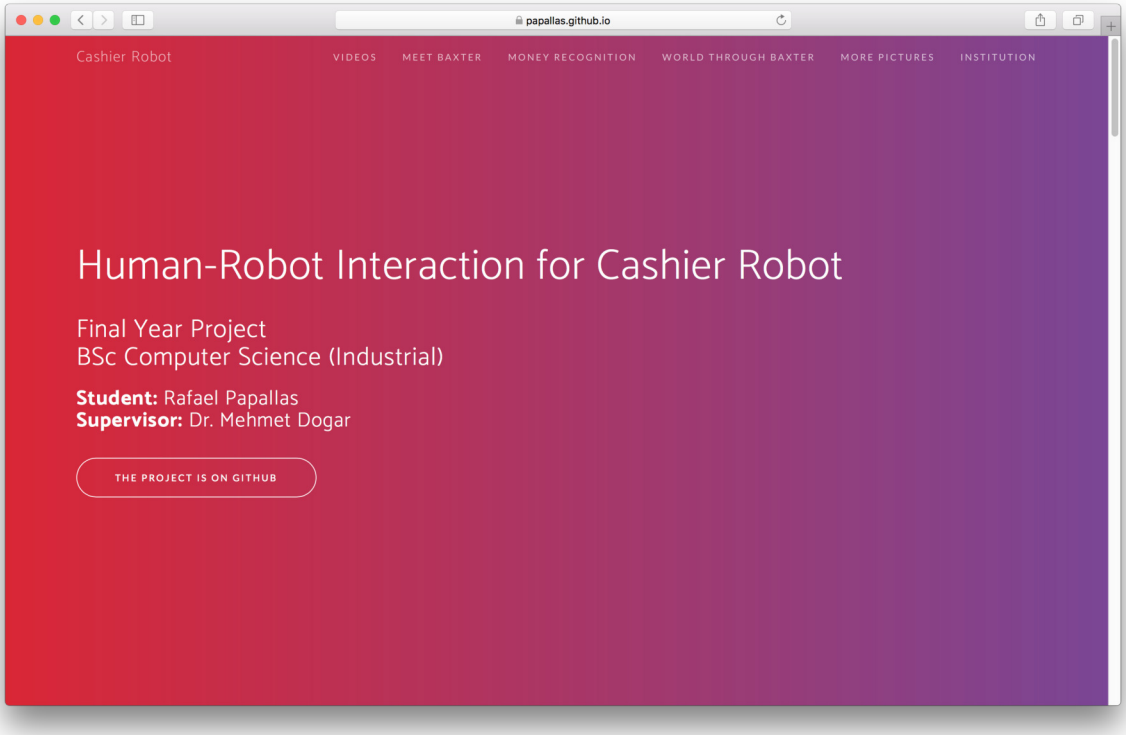


Figure 7.1: Project's website (Includes technical information about the project, videos and access to the public repository)

7.2.5 Milestones & Project Boards

To keep track of outstanding tasks to be implemented as well as having a log of completed items, GitHub issues have been used for every major and minor task of the project. These issues had a label assigned to it to indicate its purpose (Bug, Development, Research etc.) as well as milestone. Milestones reflects the three major phases of the project.

Information on how to access some of the above feature and other deliverables is available under appendix C.

[6] `https://papallas.github.io/baxter_cashier`

Chapter 8

Conclusion

8.1 Limitations

One of the limitations of the project is the manual camera calibration that requires the operator to calibrate. Because the process is entirely manual and depends on the operator, is relatively easy to calibrate the project wrongly.

In addition, the change handling is not very efficient in the sense that it requires having banknote stands to keep the banknotes vertically. This does not make the change handling too realistic but also makes harder to handle big payments since there will be a limited number of banknote on the table.

Moreover, the RGB-D camera used was the "Asus Xtion Pro Live". Unfortunately, this sensor is discontinued and hence the project depends on a sensor that is very hard to find and buy. If at any stage this sensor break or is unavailable for Baxter it means that the project is not usable since the pose detection depends on this. Moreover, the skeleton tracker has been observed that at some point and after multiple interactions stops working and requires a restart.

Additionally, although the banknotes with AR recognition is the best solution identified, when the banknotes was printed two-sided it causes some problems to the detection. There were no problems with a single-side printed banknote it only happens with a two-side printed banknote and very rarely. This seems to be a problem with a reflection from the other side and it could be easily fixed if the banknotes are printed on a more thick material.

Finally, sometimes (and after running the integrated system many times) seems like MoveIt and Baxter's default planner are conflicting. Default planners is used in last year's project, where in this project the manipulation planning is done using MoveIt. What have been observed is that when MoveIt tries to make a move it causes problems which leave the move in the middle in some cases (i.e the full move of the trajectory is not completing). This problem occurred in the very last stage of the project, and some effort was made to fix it by making the two into different processes (threads) and trying to pause the one when the other starts, although this seems to fix the problem, some times the problem is still happening. Note that this limitation is only when running the full demo including last year's project, when the project run on its own, everything works

as expected.

8.2 Future Improvements

An automatic way of determining the camera position and do the camera calibration automatically without the human needing to manually calibrate the camera would be ideal.

In addition, another possible future improvement can be the money grasping part. As of now, the money grasping is very limited since it requires the banknotes to vertically stand on the table. In the future, a better solution can be implemented that will allow the change banknotes to be placed in a "deck"-like tray and Baxter could be able to pick banknotes from the tray. This should be a very complicated task since the banknote is a very thin material, and grasping one among many should be very challenging. Moreover, if this implemented, then the money that Baxter will pick from the customer could be placed in those trays instead randomly on the table.

Additionally, when the Scoping and Planning document was initially submitted one of the main concept was an adaptive cashier. Unfortunately, due to time constraints, this feature was dropped. The idea was that Baxter can become user-adaptive so he can predict the hand-pose before is established. This feature should make Baxter much more friendly if implemented and a more realistic cashier.

Finally, from what have occurred from the participant testing and their feedback is that Baxter can make use of voice output to inform verbally the users about some events. Also, it will be much more natural if Baxter could be able to detect when the customer actually hand over the banknote (instead of having a fixed timeout where the user should give the banknote to Baxter), probably using Baxter's hand camera.

8.3 Personal Reflection

In most of the cases where problems turn up, I have chosen the solution that has the least risks. For instance, the camera calibration problem or the money grasping from the table. From this, of course, I gain experience to weight the advantages and disadvantages of different solutions and also to balance the risks.

During the development of the project many problems showed up, mainly problems that I have not thought of and in most of the cases slowed down the progress of the project. However, these problems were seen as opportunities. Opportunities in the sense that they not only taught me lessons but also they turn to be some of the most interesting topics

of my report writing. For example the calibration tool and the RGB-D sensor distance mismatch, and the three attempts for money recognition. These problems they have not let the project down but instead made the project more interesting and challenging.

When I chose this project, I knew that most of the concepts if not all were new to me. I was motivated by the idea and the outcome. Unfortunately, I have not been enrolled in the Robotics and Computer Vision modules last semester. Everything was entirely new to me and some of the time spent on the project was to explore new concepts and get familiar with terms and techniques required for a robotic project. If I was enrolled to those modules, I would be able to predict some of the problems that appeared during the development of the project. On the other hand, this unique experience of not feeling secure about the topic with the combination that the project will determine my degree class had some positive impacts to myself. Not feeling confident about the topic was one of the best experience I have faced in the university so far, and at the end was one of the critical points for the project's success.

If I was to do the project again, I would make a more realistic schedule that would allow time to do the important parts of the project with more attention and eliminate mistakes that lead to a choice of a basic solution like the money grasping from the table.

I enjoyed every piece involved in this project, from reading about the new concepts of robotics to designing and implementing of several solutions to the report writing. My supervisor helped me to focus and think about some other issues that I was not able to see at the time. Most of the times, I was motivated about impressive and complicated tasks or solutions that were not too realistic, but my supervisor taught me to think about the time as well, and that the solution should be realistic given the time left and always try to implement something basic, and if time left evolve it.

In addition, I have learned how a research project is done through experience, from background and literature review, design, experiments, development and testing. This experience should be very helpful for my future career plans.

Finally, I knew that a big challenge of this project is the time management. I was well prepared to face this challenge by being well organised, maintaining to-do lists and agendas at every meeting as well as getting things done on time and eliminate procrastination most of the time.

References

[1] Rethink Robotics. About ROS, 2017. URL `http://www.ros.org/about-ros/`.

[2] Rethink Robotics. ROS Indigo Igloo, 2017. URL `http://wiki.ros.org/indigo`.

[3] Krystof Litomisky. Consumer rgb-d cameras and their applications. *Rapport technique, University of California*, 20, 2012.

[4] Derek Hoiem. How the kinect works, 2011. URL `https://courses.engr.illinois.edu/cs498dh/fa2011/lectures/Lecture%2025%20-%20How%20the%20Kinect%20Works%20-%20CP%20Fall%202011.pdf`.

[5] S. Corazza, L. Mündermann, A. M. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34(6):1019–1029, 2006. doi: 10.1007/s10439-006-9122-8.

[6] Arnaud Ramey, Victor Gonzalez-Pacheco, and Miguel A Salichs. Integration of a low-cost RGB-D sensor in a social robot for gesture recognition. In *the 6th international conference*, pages 229–230, New York, New York, USA, 2011. ACM Press.

[7] Christian Ott, Dongheui Lee, and Yoshihiko Nakamura. Motion capture based human motion recognition and imitation by direct marker control. In *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pages 399–405. IEEE, 2008.

[8] Liying Cheng, Qi Sun, Han Su, Yang Cong, and Shuying Zhao. Design and implementation of human-robot interactive demonstration system based on kinect. In *2012 24th Chinese Control and Decision Conference (CCDC)*, pages 971–975, May 2012. doi: 10.1109/CCDC.2012.6242992.

[9] Chien-Ming Huang, Maya Cakmak, and Bilge Mutlu. Adaptive coordination strategies for human-robot handovers. In *Robotics: Science and Systems*, 2015.

[10] Zhengyou Zhang. Microsoft Kinect Sensor and Its Effect. *IEEE Multimedia*, 19(2): 4–10, 2012.

[11] Vincenzo Micelli, Kyle Strabala, and Siddhartha Srinivasa. Perception and control challenges for effective human-robot handoffs. In *RSS 2011 RGB-D Workshop*, June 2011.

[12] Krishnanand N. Kaipa, Carlos W. Morato, Jiashun Liu, and Satyandra K. Gupta. Human-robot collaboration for bin-picking tasks to support low-volume assemblies. 2014.

[13] H Reddivari, C Yang, Z Ju, P Liang, Z Li, and B Xu. Teleoperation control of Baxter robot using body motion tracking. *MFI*, 2014.

[14] Rethink Robotics. Hardware specifications, 2015. URL `http://sdk.rethinkrobotics.com/wiki/Hardware_Specifications`.

[15] Rethink Robotics. Cameras, 2017. URL `http://sdk.rethinkrobotics.com/wiki/Cameras`.

[16] John J Craig. *Introduction to robotics*. Addison-Wesley, 2 edition, 1989.

[17] Ioan A. Sucan and Sachin Chitta. Concepts | moveit!, . URL `http://moveit.ros.org/documentation/concepts/`.

[18] Ioan A. Sucan and Sachin Chitta. Moveit! motion planning framework, . URL `http://moveit.ros.org/`.

[19] Rethink Robotics. Moveit tutorial - sdk-wiki, 2015. URL `http://sdk.rethinkrobotics.com/wiki/MoveIt_Tutorial`.

[20] Vladimir Chernov, Jarmo Alander, and Vladimir Bochko. Integer-based accurate conversion between rgb and hsv color spaces. *Computers & Electrical Engineering*, 46:328 – 337, 2015. ISSN 0045-7906. doi: http://dx.doi.org/10.1016/j.compeleceng.2015.08.005. URL `http://www.sciencedirect.com/science/article/pii/S0045790615002827`.

[21] Rodney Brooks. Rodney brooks: Why we will rely on robots. URL `https://www.youtube.com/watch?v=nA-J0510Pxs`.

Appendices

Appendix A

External Materials

The project makes use of some external libraries to achieve the end-result. These libraries are used mainly because they are well-tested and developed and they offer a very robust outcome. The libraries are highlighted below:

- `cob_people_perception` (`https://github.com/ipa-rmb/cob_people_perception`). This library provides the skeleton tracker and makes possible to detect the user's hand-pose.
- `ar_track_alvar` (`http://wiki.ros.org/ar_track_alvar`). This library provides the banknote recognition by recognising AR codes as well as it provides a mean to generate AR codes.
- A forked version of "Calibration Tool". The calibration tool that is used to solve the problem of robot-camera mismatch is based on an initial solution developed by Muhannad Al-Omari (`https://github.com/papallas/baxter_cashier/wiki/Camera-Calibration`).
- The project also make use of "MoveIt!" the popular framework that provides collision-aware planner (`http://moveit.ros.org/`).
- Other libraries like `ROS`, `OpenNi2` and `OpenCV` are also used in the project.

Appendix B

Ethical Issues Addressed

Data Privacy

The project does not keep or store locally or on the cloud any pictures or data of the users.

Employment

This project is real-life applicable. Someone will argue that such project could have a negative impact on our society since robots can replace people in the supermarkets.

The intent of the project, however, is to create a friendly and entertaining demo for the open days at the University of Leeds as well as to explore and implement algorithms for human-robot interaction and collaboration.

A very interesting speech by Brooks highlights that the demographics show that many jobs that need doing will not be fulfil by our society and instead we will need to rely on robots [21].

Furthermore, Brooks also mentions the case with librarians in 1957. In specific, he highlights the case where mainframe computer was brought to the librarians to help them with their job, but the librarians were afraid that they will lose their jobs. Brooks shows that the opposite happened. In 1957 there were sixty thousand librarians and in 2009 more than two hundred thousand librarians. This case tells us that the invention and use of computers, artificial intelligence and robots in the industry does not close jobs but instead opens new job opportunities, especially for the new generations.

Finally, many supermarkets in the UK and around the globe use similar machines for self-checkout, although they are not humanoid robots but just machines with barcode readers and money recognition. With that said, this project does not bring a new application of robots in the supermarkets as they already exist for years and they are used to speed up the queues at the supermarkets although real cashiers are not replaced by these machines. Moreover, such machines require humans to supervise the process.

Appendix C

Access to other deliverables

This appendix is to list some other deliverables and useful links related to the project. Shorten URLs were generated to make access to the resources easier. If however some of the shorten URL does not work, reference to the full URL provided as a footnote.

1. Project's repository (GitHub): `http://bit.ly/cashier_repo`[1]
2. Project's Wiki pages: `http://bit.ly/cashier_wiki`[2]
3. Project's website: `http://bit.ly/cashier_website`[3]
4. Youtube video with voice over: `http://bit.ly/cashier_vid_1`[4]
5. Youtube video (demo): `http://bit.ly/cashier_vid_2`[5]
6. Modified[6] last year's project repo (GitHub): `http://bit.ly/shopkeeper_repo`[7]

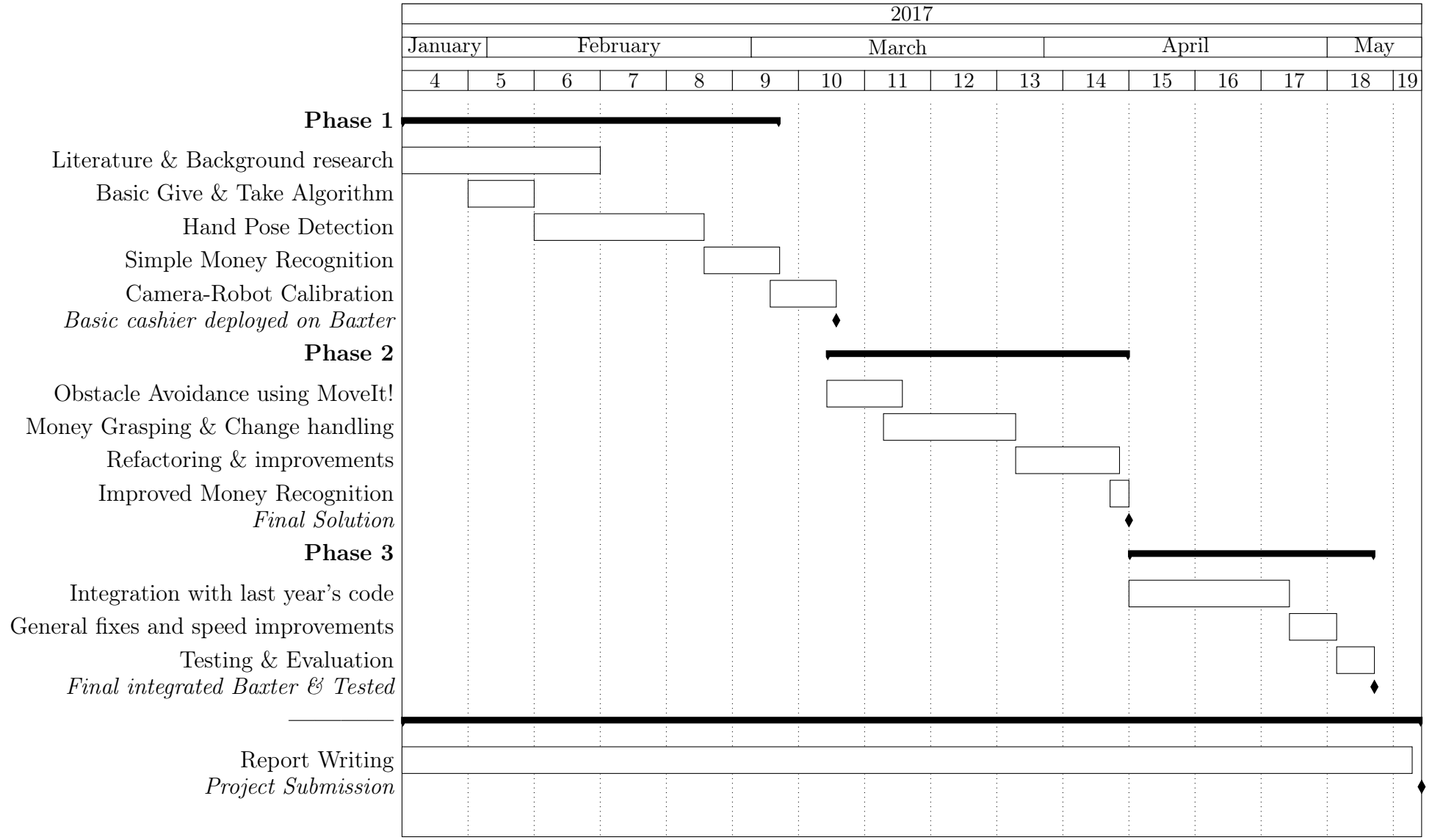
[1] Project's repository full URL: `https://github.com/papallas/baxter_cashier`
[2] Project's wiki pages full URL: `https://github.com/papallas/baxter_cashier/wiki`
[3] Website full URL: `https://papallas.github.io/baxter_cashier`
[4] Youtube video with voice over full URL: `https://youtu.be/3DlCj9DgiZA`
[5] Youtube video (demo) full URL: `https://youtu.be/dUGXIO3NPXM`
[6] Changes made to do the integration
[7] Modified last year project full URL: `https://github.com/papallas/baxter_shopkeeper`

Appendix D

Gantt Chart

Below is the project progress in Gantt chart. Please note that the Gantt chart was constructed based on the meeting agendas maintained during the weekly meetings with the supervisor, which contains approximately the duration of the tasks. With that said, this is just a very close approximation to the actual progress but not the exact progress in some cases.

The Gantt chart is split into year weeks and not into academic weeks. The Gantt chart starts from the 4th week of January (23rd of January 2017) which was the official start of the second semester. The main reason that the Gantt chart was split into year weeks is to also include the work done during the Easter break.


2017
January
February
March
April
May
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Phase 1
Literature & Background research
Basic Give & Take Algorithm
Hand Pose Detection
Simple Money Recognition
Camera-Robot Calibration
Basic cashier deployed on Baxter
Phase 2
Obstacle Avoidance using MoveIt!
Money Grasping & Change handling
Refactoring & improvements
Improved Money Recognition
Final Solution
Phase 3
Integration with last year's code
General fixes and speed improvements
Testing & Evaluation
Final integrated Baxter & Tested
Report Writing
Project Submission

Appendix E

Risk assessment

There are some risks that may or may not happen during the design, development or/and testing of the project. These risks are highlighted in **Table E.1**.

Table E.1: Risk Assessment

Risk	**Probability**	**Impact**	**Alternative strategy or solution**
Robot Damage	Unlikely	Major	Simulator
Participants injury	Unlikely	Major	Safety introduction
Banknote grasping failure	Likely	Major	Separation of banknotes
RGB-D sensor not integrating	Likely	Minor	Use other RGB-D sensor
RGB-D sensor too noisy	Likely	Minor	Motion Capture System

Robot Damage

Baxter as any hardware system can be damaged or can have hardware failures. Since the project depends on this specific hardware, failure will be a risk for the project. School of Computing has only one such robot available which makes the risk a major one. On the positive side, ROS and Rethink Robotics provides an alternative solution. Use of the Gazebo Simulator.

Participants injury

The project will require some participants to do testing or for data collection. Since the participants will interact with the robot if they are not informed about general safety rules they may injury themselves.

Introduction about how Baxter will behave and the ways they can injury themselves as well as supervision of experienced person and usage of the e-stop[1] button will be enough to avoid this risk.

Banknote grasping failure

Banknotes will be printed on A4 papers. A4 paper is a very thin material and grasping a single banknote from a stack of banknotes is very challenging.

A possible solution is to separate the banknotes into single units and made them available to the robot either by leaving their edge outside the table surface or using some placeholder

[1]e-stop is the physical red button near Baxter that stops Baxter immediately

to vertically lock the banknote to a fixed position.

RGB-D sensor not integrating

RGB-D cameras are manufactured by a variety of companies. Most of the companies do not make the drivers open source and hence a volunteer community takes the responsibility to port the driver for ROS. Recent and new sensors (e.g Microsoft Kinect One) may luck open source and ROS support. In such case, use of a more popular sensor will be used.

RGB-D sensor too noisy

One of the RGB-D disadvantages is that they are in most of the cases noisy. Since in this project, accuracy to millimetres is not important this will be a low probability risk. If however the sensor is too noisy and it causes problems to the project, use of Motion Capture System will be considered.

Appendix F

Changes since Scoping and Planning document

Changes

Some changes have occurred since the submission of the Scoping and Planning document[1].

In specific, the original plan was to implement a basic cashier for phase 1, and then phase 2 would be delegated to make the robot adaptive, so it can act differently from customer to customer. Moreover, the money recognition was said to be a basic solution since the attention would be on adaptivity.

During the development of the project, some problems showed up with the camera driver and skeleton tracker which added much more delays to the schedule than initially planned as well as the problem with the camera pose calibration which was not on the schedule at all. Since some critical problems showed up, it was fair to make some changes to make the schedule realistic given the problems that have shown up.

Adaptivity

Adaptivity, in this project, was a feature described in the Scoping and Planning document, where classification algorithms would be implemented to classify the posture of the users so it can predict which hand to approach. For example, if the user's left hand is clearly more close to the cashier than his right hand, the robot will reach his left hand. These will require constructing an initial training set from participants and use this training set to train the classification algorithms.

Adaptivity was one of the main features highlighted in the Scoping and Planning document. However, this feature has been cancelled. The time required to record data from fifteen participants with the combination of writing classification algorithms and making sense of the data made this feature unrealistic. The decision of dropping the adaptivity part over other functionalities, was mainly because the adaptivity would not have any impact to the end-result.

[1] Scoping and Planning document was a formal initial plan submitted at the beginning of the semester to receive early feedback.

Money recognition

In the Scoping and Planning document, money recognition was of secondary priority since the adaptivity part was of higher priority. Now that the adaptivity part is not anymore in the objectives, this allowed some more time to consider a better solution for money recognition and make the money recognition a primary objective.

Gantt Chart

It turns out that the Gantt chart created for the Scoping and Planning document was not too realistic since it did not consider some main tasks like money grasping from the table. In Scoping and Planning document Phase 2 was delegated to the adaptivity part. Phase 2 has now been replaced with several improvements and also money grasping from the table as well as integration of obstacle avoidance planner ("MoveIt!").

The revised version of the Gantt chart is available under Chapter D.

Final thoughts on the changes

In conclusion, although a major feature like adaptivity was dropped, the time has been used much more efficiently and led to a more robust overall outcome since money recognition took more attention and was implemented better. Moreover, the time left was also used to make Baxter collision-aware[2], which is much more important than having an adaptive cashier that would possibly cause harm to itself.

[2] Baxter is aware of obstacles

Appendix G

Participants feedback form

Human-Robot Interaction for Cashier Robot
Participant Evaluation Form

Interaction took: _____ minutes

1. From 1 to 5, how did you found the interaction with the robot?

   (Bad) 1 2 3 4 5 (Excellent)

2. From 1 to 5, how similar was the interaction with the robot cashier with a real cashier?

   (Not Similar) 1 2 3 4 5 (Very Similar)

3. From 1 to 5, how clear was the process to you? (i.e You understood when you had to wait for the robot to do something?, you understood when you had to take action? etc)

   (Not Clear) 1 2 3 4 5 (Very Clear)

4. Was the positioning of robot's arm accurate and of your convenience when the robot approached your hand(s) during the interaction?

   Yes No Not Always

5. How would you rate the speed of the robot during the interaction?

   Slow Normal Fast

6. Do you think this project can be improved? If so, please write few words as a feedback.

_____

_____

_____

_____

_____

_____

Would you be interested to include a picture of the participation in the final project report?

Yes No

_____ _____

Participant Full Name Participant Signature