MTRN 4230 - Assignment 1 Management Report

Affable Automatons: Group 5
Vivek Desai , Niels Alston , Adam Kelly-Mills , Roshen Mathew Roshen Mathew , Ting Mei , Xiangfen Y

June 3, 2016

Contents

1 Introduction

The following document outlines the tasks assigned to individual group members in the aim to complete the criteria required of assignment 3. This will take the form of an outline of what was required of each member or "team" at each lab due at the start of the next.
Communication of these requirements was done through an on-line project management software called trello as well as weekly in lab meeting at the start of each lab.
Code was managed and consolidated using the code management software/website GitHub with all members having access to the hared repository.
Google Hangouts was used as a medium for more informal discussions that did not require documentation.

2 Management Plan

2.1 Schedule

The assignment completion deadline allowed for a 3 week time frame for work where the IRB120 robot was available once a week for 3 hours at a time.

Given these restriction on availability with the robot a time schedule was required to ensure that sufficient time was available for testing of developed code for interfacing with the live robot.
The overall assignment was broken up amongst the team members in individual groups to work on relavent Tasks

During The first meeting following the individual submissions of Assignment 2 Most of this time was spent fixing up group Assignment 1 code as well as creating a task breakdown for and understanding assignment 2. The team was broken up into 3 groups - Robot Studio team (Roshan and Linda), Matlab Graphical User Interface (GUI) team (Niels and Xiangfen)and the image recognition/handling team (Vivek and Adam). The following tasks were assigned in each meeting to the teams

2.1.1 Meeting 1

Robot Studio Team Improve communication between matlab and robot studio to reduce operation time.

GUI Team Create a outline plan / wireframe of the required GUI for Assignment 3 working off the assignment 1 GUI.

Image Recognition Team Begin Creating a more comprehensive template for surf point comparison and determine oporations required to convert image data into metrics used in robot studio.

2.1.2 Meeting 2

Robot Studio Team Creating of Robot Studio Processes so as to allow for single command interfacing with Matlab through the Matlab GUI. Fix Singularity Cases or create fail safes for escaping singularities

GUI Team Create a outline plan / Development of Matlab GUI as well as the development of simple control flow charts for the operations required for each Assignment criteria.

Image Recognition Team Refinement of image recognition functions, a key issue was the excessive time required to perform image processing on a single workspace image - test the implementation of image subtraction to remove unnecessary clutter from the workspace image.

2.1.3 Meeting 3

At this meeting many teams worked collaboratively in order to enable interfacing of their work with the overall project.

Robot Studio Team Testing of GUi Interface with Robot Studio.

GUI Team Implementation of Image Recognition as well as operational control sequences such as stack management and loading/unloading control flow.

Image Recognition Team Calibration of image detection with that of the conveyor belt and adjusting of image detection function to also enable box recognition. It was also required that image recognition evaluate stack locations within the box for loading.

3 Code Development and Basic Theoretical Principles

3.1 Graphical User Interface and Function Flow

The first step for completing the GUI component as mentioned was the design of the GUI. We opted for a tabular approach which helped de-clutter the screen and make the interface more intuitive. The tabs were split into a Debug Tab, for manual jogging and other similar debugging options. A Sorting tab, which only involved actions which required a single button press such as sorting the table and moving chocolates to predefined positions then finally the Order Management Tab which processed orders.

3.1.1 Debug Tab

The Debug Tab inherited the functions in the GUI for assignment 1, for carrying out single actions to test the connection to the robot, which includes the video feed, moving of joints of the robot arm, turn on, turn off and change running direction of conveyor belt, turn on and off the vacuum power and solenoid.

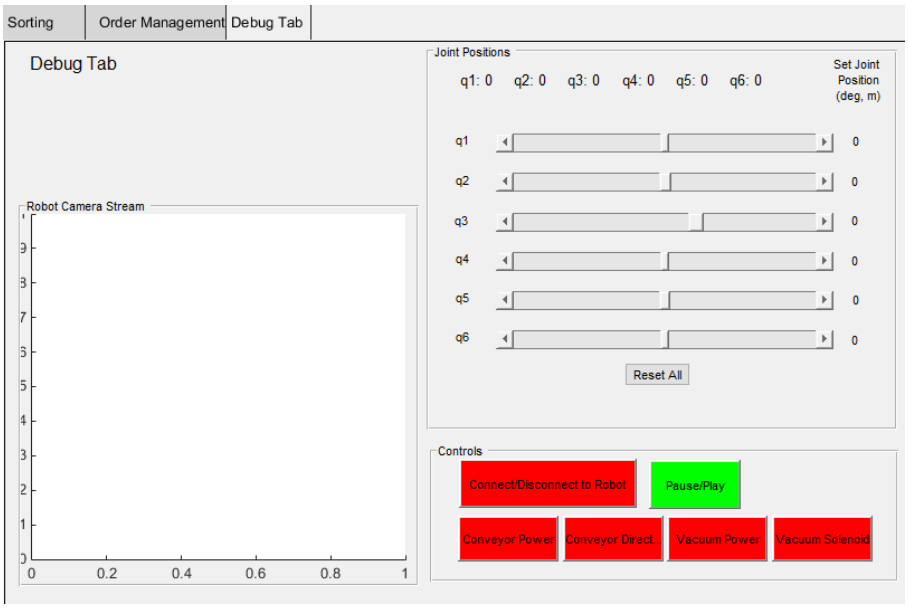

Figure 1:

3.1.2 Sorting tab

The Sorting Tab contains the functions: Display the camera video feed, send action instructions to Robot Studio including place a single chocolate in given position, sorting the chocolates on the table with given order, unload the chocolates in the box on conveyor belts which corresponds to the requirements in the Assignment specification sheets; and a button for pausing and playing the current instruction execution.

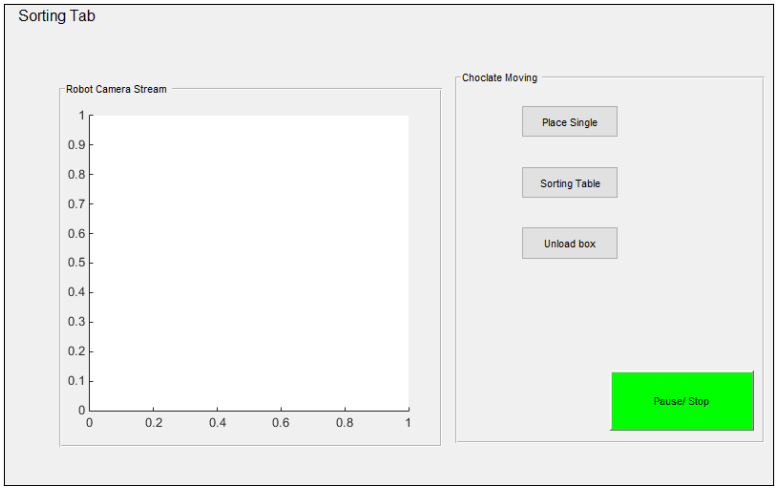

Figure 2:

3.1.3 Order Management Tab

The Order Management Tab contains the functions: Let user give the order shown in a stack by click the buttons in the order stack which contain the position and type information of the desired chocolates, then the buttons for sending instructions to fulfill the order when the order is checked to be able to fulfilled with the available chocolates on the table, then a button for reset the order stack to empty and a button to send the action instruction for sending the loaded chocolate box on conveyor belt. And button for pausing or playing the execution of these instructions and a message box for showing the messages.

Figure 3:

The Code Flow Diagrams can be represented as shown below in figure 4

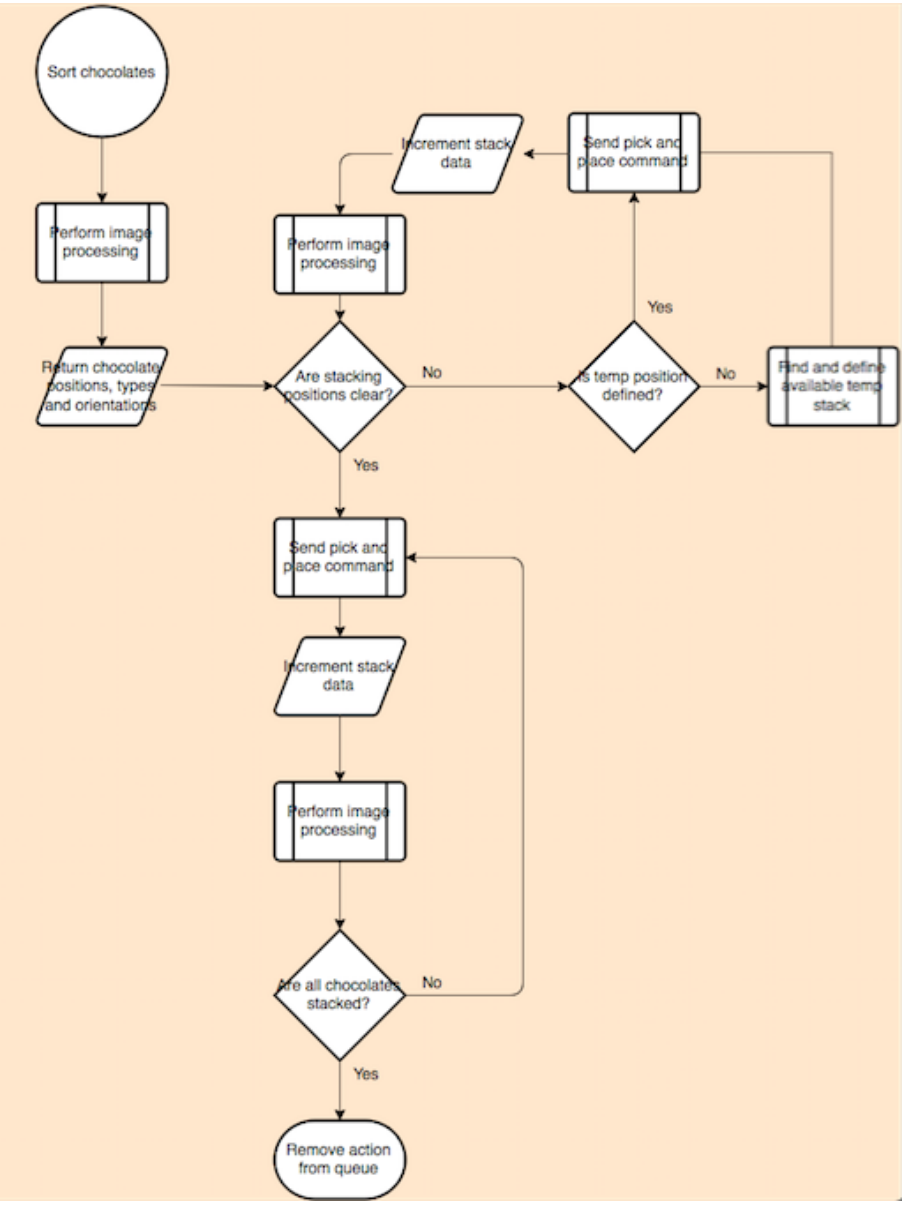

Figure 4:

3.2 Image Detection

The code for Image Processing is fairly distributed. The folder 'ImageRecognitionStuff' contains calibration and template images, as well as collated template data and assorted functions. These functions include CreateTemplateData, which was used in conjunction with cropping.m to gather template data for each individual type of chocolate, as well as FeatureMatching.m, which performs the actual matching between chocolates and the image feed. Extractdata.m checks chocolate reachability, and assigns each chocolate a boundary and centroid. To improve image processing quailty, images of the empty conveyor and table (as shown below) were used to isolate the chocolates from the background, through the direct subtraction of image data.

3.2.1 Image Subtration

Figure 5:

By subtracting the background information, image processing can be faster and simpler to operate.

3.2.2 Metric Extraction

Of note are DetectBox.m and DetectChocolates.m, which use feature matching functions to generate locations for both the box, and arbitrary chocolates detected by the camera during the image processing phase. These functions are all called from within z3373202 MTRN4230 ASST2.m, which changes the base frame to that of the robot, and produces an object called ChocolateData. Properties of this object are used by many functions within ABB GUI Automated.m, as it is the only source of data for chocolates to be recognised by the system.

3.2.3 Stacks

Stacks are a pile of chocolates in the same location, which necessitates a slightly different formalism in their operation, to account for variations in type and height for each chocolate in the stack. Stacks are primarily defined and manipulated using the functions found in TestStack.m, and contain individual data on the type and order of each chocolate in the stack. This may be contrasted with ChocolateData, which is more compact, but assumes each chocolate is simply placed on the table. StackingPosClear.m and defineTempStack.m help to produce behaviour that allows the robot to clear the general stacking area (defined as a box one half-chocolate-length around the individual stacking positions for each chocolate type, in order to reduce collisions). StackingPosClear checks ChocolateData to find chocolates in this region, and defineTempStack places a temporary stack on top of a chocolate outside this region, in order to keep track of chocolate displacement. If no chocolate is found outside this region, defineTempStack defaults to using an empty, predefined position for this temporary stack.

3.3 Interface with IRB120 through ABB Robot Studio

Rapid Studio The code written in Robot Studio mainly involves manipulating the robotic movement and systems in the work cell in defined procedures and communicating with MATLAB GUI.

3.3.1 Communication between Robot Studio and MATLAB GUI

For Robot Studio to understand the request string message, the message is parse into the eleven sections using the strPart function. The following sections are then converted into number variables to manipulate the robot into the correct pose: joint angle value, the three components of the initial position and orientation of the chocolate and three components of the final position and orientation of chocolate.
After decoding the message, the required actions and corresponding functions or procedures is determined through conditional branches.

3.3.2 Systems in the Work Cell and Robot Manipulation

The major priority of the robot manipulation is to pick up chocolate at the given initial location and set down the chocolate at the required destination. Due to the desire of singularities in the robot pose being avoided, as well as keeping the code and movement to be simplistic, five set of procedures are written and labelled in accordance to the initial location and final destination of the chocolate in the work cell. The five procedures are called

Table to Table Movement

Table to Flip Movement

Flip Retrieval to Table Movement

Table to Conveyer Movement

Conveyer to Table Movement

3.3.3 Protocols for Communicating between MATLAB and Robot Studio

For controlling the robot arm with a graphical user interface made in MATLAB, a client/server based protocol style, using a pair of request and response message, is adopted for communication between MATLAB and Robot Studio. The client, i.e. MATLAB, sends a request message (usually an action) to the server, i.e. Robot Studio, and the server respond back with a message after processing the request. The style of the request and response messages is as follows.

MATLAB Client Requests The request message is a string comprised of eleven sections: action code, conveyer/vacuum mode, joint angle value, the three components of the initial position and orientation of the chocolate and three components of the final position and orientation of chocolate. The order of the sections is shown in the below message template example and each section is separated by a hash.

Message Template Example
String message = [Action] # [Conveyer/Vacuum] # [Joint Angle] # [Initial X] # [Initial Y] # [Initial Z] # [Initial Theta] # [Final X] # [Final Y] # [Final Z] # [Final Theta]

The action code is comprised of 3 characters where the first character is S followed by the action number. There is a total of 17 actions and these can be seen in table 1. There is 1 character for the running of conveyer and vacuum processes, itll either be a 0 or 1. 7 characters are used for each of the following: joint angle, initial x, initial y, initial z, initial theta, final x, final y, final z, final theta. First character is for the sign, second to fourth character are the digits before floating point, fifth character is the floating point, and sixth to seventh are the digits after floating point. The total number of characters used in the string message is 78. Sections that arent required in the command is filled with the character X.

Table 1: Action Code and Associated Movements

Action number	Action
1	Joint 1 Jog
2	Joint 2 Jog
3	Joint 3 Jog
4	Joint 4 Jog
5	Joint 5 Jog
6	Joint 6 Jog
7	Conveyer Power
8	Conveyer Direction
9	Vacuum Power
10	Vacuum Grip
11	Pause/Play
12	Table to Table Movement
13	Table to Flip Movement
14	Flip Retrieval to Table
15	Table to Conveyer
16	Conveyer to Table
17	Out of Camera View

3.3.4 RAPID (Server) Responses

The response message is a string message categorised into three classes: success, client error and server error. Please refer to table 2 for the description of the class types. The string message is usually a code composed of three number characters (see table 3 to table 5). The exception is when Robot Studio is responding to a request for an aspect of the robot workstation (i.e. get action). For a successful get action, the string sent by Robot Studio should be 100/Space/ NUM STRING where NUM STRING represents the corresponding value or name to the requested. Response Example for a get Request a) The client request the status of the jogging mode joint3 orientation and Robot Studio has successfully read the joint3 orientation value of 46.7 degree. Then the response is a string = 100 46.7

Appendix 1 : Requirements Tracking

RibStudi		23/05/2016		Roshan,Linda		co-ordinates	multitasking		vacume/conveyor
New GUI		23/05/2016		Xiangfan,Niels		Debug Tab	Sorting Tab		Order Management
Image		23/05/2016		Vivek,Niels		using surf point - refine	Image subtraction		box recognition
Code		23/05/2016		Vivek,Linda				Box image detection	Vivek
Requirem	Marks	Status	Date last	Responsible (group member)	Tested by (group	Title	Description		Comments
Picking									
1.1	1	Pass.	20160416			**Pick up easy chocolates**	Picking up chocolates where none overlap and all are reachable.		
1.2	1	Pass.	20160416			**Pick up overlap chocolates**	Picking up all initially pickable chocolates where some overlap initially (up to 50% of visible area of a chocolate).		
1.3	1	Pass.	20160416			**Pick up all chocolates**	Picking up all reachable chocolates, including correctly removing pickable chocolates before those hidden underneath		
1.4	1	Pass.	20160416			**Flipping all inverted chocolates**	Flipping all reachable inverted chocolates so they face up		
Stacking									
2.1	1	Pass.	20160416	y		**Place one chocolate**	Place a single chocolate in the given centre location and orientation		
2.2	1	Pass.	20160416	/		**Place one chocolate of each flavour**	Pick and place one of each kind of pickable chocolates in the following locations		
2.2	1	Pass.	20160416	/		**Stack all chocolates**	Neatly stack all visible chocolates into four piles with positions defined in the previous sub-section, accounting for the		
Load/Unlo									
3.1	1	Fail. Solution	20160416			**Load 1 chocolate into box**	Load one chocolate into a box on the conveyor, regardless of position or orientation		
3.2	1	Fail. Solution	20160416			**Load 4 chocolates into box**	Load four chocolates into the four positions (B1 to B4) within a box on the conveyor so they all lie flat without overlap		
3.3	1	Fail. Solution	20160416			**Unload 1 chocolate from box**	Unload one chocolate onto the table from a box on the conveyor		
3.4	1	Fail. Solution	20160416			**Unload all chocolates from box**	Unload all chocolates onto the table from a box on the conveyor, where there will be at least 5 chocolates in the box,		
Basic									
4.1	1	Fail. Solution	20160416			**Load one type of chocolate into box**	To select one kind of the four kinds of chocolate and the number to be loaded into a single box		
4.2	1	Fail. Solution	20160416			**Load multiple types of chocolates into**	To select multiple kinds of chocolates and the number of each to be loaded into a single box		
4.3	1	Fail. Solution	20160416			**Load pattern of chocolates into box**	To manually specify the pattern of chocolates to be loaded into a single box.		
4.4	2	Fail. Solution	20160416			**Pause/stop and recover**	To enable the motion of the robot to be stopped (order cancelled), paused (order maintained), restarted and to recover		
Advanced									
5.1	1	Fail. Solution	20160416			**Possible to fulfil order?**	Identify if it is possibly to fulfil the order and request a new order if it is not. You may need to unpack or flip chocolates		
5.2	5	Fail. Solution	20160416			**Fulfil order**	Fulfil any order (as per the previous section) once the user has confirmed the pattern of chocolates. Display the current		
Document									
6.1	2	Pass.	20160416	Vivek		**Documentation**	All of your functions must be well documented within the code in both Matlab and RAPID. Include at least a Help		
6.2	2	Pass.	20160416	Vivek		**Requirements tracking**	Make a requirements tracking spreadsheet of your own, and submit an electronic copy to your demonstrator in your lab		
6.3	1	Pass.	20160416	Vivek		**Code management**	The current status of your code management practices will be queried by your demonstrator each week. You need to		
6.4	3	Fail. Solution	20160416	Vivek		**Management report**	You will need to provide a short report (<10 pages) detailing your project management practices and giving evidence of		
Total	30								

Figure 6: