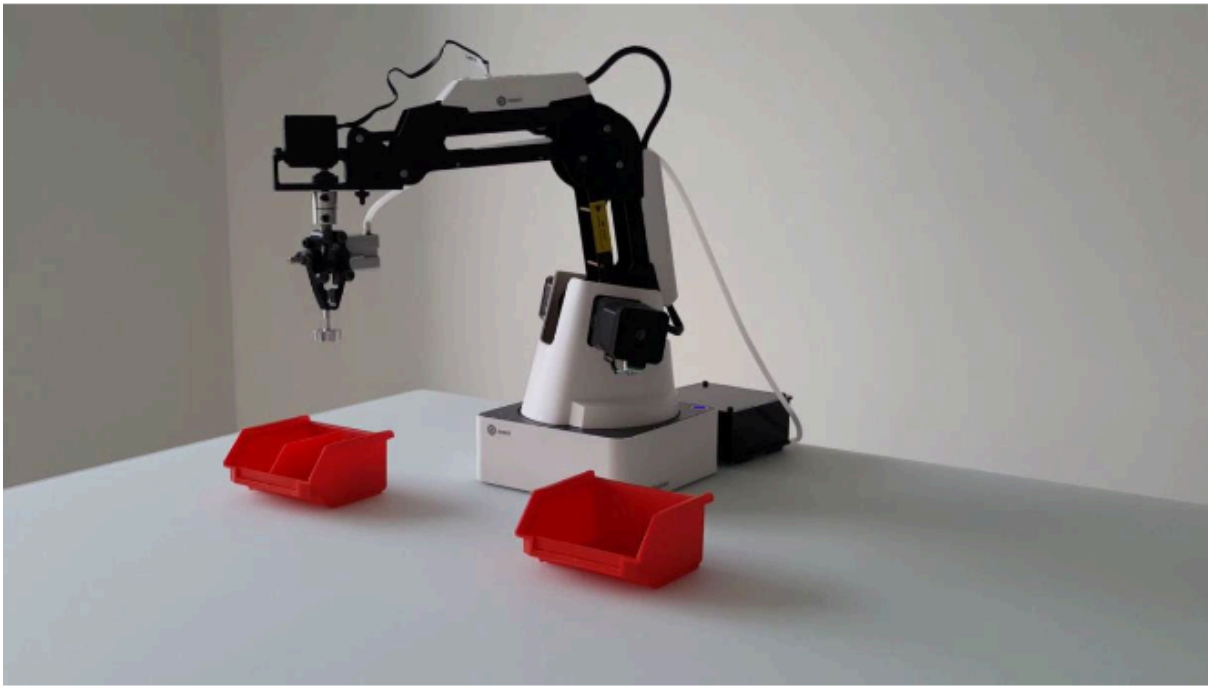
DoBot Robot - Project 3



Group 38

Connor Mawer, Cassandra Cecconi, Ash Mudaly, Lachlan Scott Rogers

Table of Content

1.0 Introduction

The following report will cover all the necessary steps and software required to recreate Project 3 - for picking and placing an object - with the specific objective of arranging the objects into an arrangement. This project aims to develop a robotic system that automatically detects, and grasps objects. This will be accomplished by using an RGB-D sensor through a real sense camera and MATLAB for controls, this system will enable the DoBot Magician to interact with the objects in a controlled workspace.

1.1 Objective

The main aim of this project is to develop a robotic system that autonomously picks and places objects of varying characteristics (such as; colours, shapes and size). More specifically, the system will use ROS for communication and calibrations of the RGB-D sensor and MATLAB for image interpretation, object detection and controlling the DoBot Magician. Project 3 requires real-time communication to integrate both systems, to allow the DoBot robot to pick and place the objects into specific formations.

1.2 Group-defined Scope

Project 3 demonstrates a practical application of sensors and robotic controls in terms of picking and placing. The DoBot robot, equipped with a suction cup, identifies and moves objects within a workspace, sorting them into predefined arrangements/positions. Through controlled actions, this setup mimics real-world scenarios such as automated sorting and organisation, and fine motor skills, plus the opportunity to utilise the program for teaching and learning, specifically as a lesson in the occupational therapy (OT) sector.

1.3 Team Members and Roles

We ensured the workload was distributed evenly across all team members according to individual strengths. Team members with a background in Industrial Robotics focused on the DoBot controls and system integration, while others are responsible for camera celebration, image processing, and documentation. Every team member contributed ideas and feedback throughout the entirety of the project, and team members collaborated to ensure all deadlines were met. Additionally, stepping in to support each other if someone was unable to complete their work by the scheduled date.

2.0 System Overview

The following section provides details regarding the hardware and software utilised to complete project 3 - picking and placing. In particular, it will discuss the purpose and role of each component in achieving the project objective.

2.1 Hardware Setup

Enabling the group to undertake the project required the utilisation of several hardware components, which are further explored in section 2.1 Hardware Setup. Together, these components create an integrated system for object detection, localisation, and picking and placing operations.

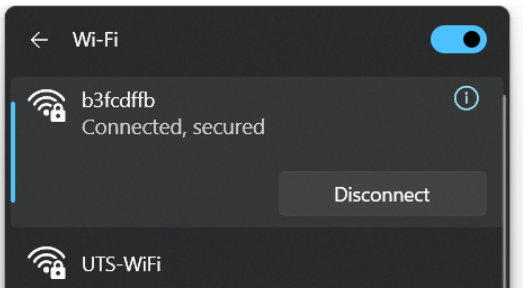
2.1.1 DoBot Magician Robotic Arm

The DoBot magician is a 4 DOF (degrees of freedom) multi-functional robotic arm designed for educational and research applications (DOBOT Magician, n.d.). The DoBot was selected for its high-precision programmable controls and ease of integration with MATLAB and ROS. Sourced from the public repository (provided in Industrial Robotics), the DoBot libraries enable the practice of the DoBot arm manipulation. The DoBot's end effector is equipped with a suction cup to allow the picking and placing of the coloured counters in the predefined area.

2.1.2 RGB-D Camera (Intel Realsense)

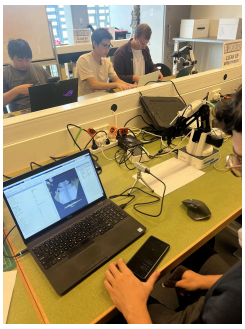
The Intel RealSense camera was chosen for its stereo depth capabilities, which provide real-time data, and accurate object localisation (Intel®, 2024). for applications like robotics, 3D scanning, AR/VR, and visual servoing. With capabilities to accurately sense depth from 0.3 to 10 metres, and a field of view spanning 87.2° horizontally and 58.9° vertically, the RGB-D camera takes advantage of GPU and USB device support for efficient real-time image processing on Windows and/or WSL; which is essential for guiding the DoBot to the correct location for picking and placing of the counters (Ademovic, n.d.).

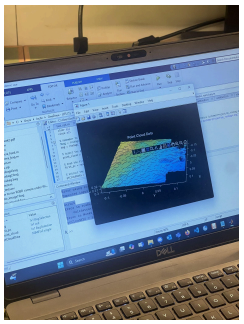
2.1.3 Laptop and Raspberry Pi



A Windows 10/11 operating system is required to operate the system's core functions. It should be noted that running WSL tasks on Apple Silicon **does not work**, which will be explained further in 4.0 Results and Testing. Connecting to the DoBot magician required a Raspberry Pi, with the appropriate firmware flashed, to allow communication via Wi-Fi. This itself had its issues; see 4.0 for more details.

2.1.4 Camera Calibration and Initial Setup





Camera calibration and setup involved following the steps outlined in "How to run RGBD camera under Windows Subsystem Linux.pdf". Ensuring that the camera is attached to the USB port, ROSbags and live footage can be received and interpreted, see images below. ROS' inbuilt calibration tools allowed us to accurately map the workspace and define object locations (Yuejiang, 2019).

2.2 Software Overview

2.2.1 MATLAB

MATLAB facilitates image processing, including RGB to HSV colour conversions, which enables object detection based on user-defined inputs of option arrangements (which define how colours are arranged). MATLAB was also utilised for the Dobot control.

2.2.2 ROS Communication (Robot Operating System)

ROS is a middleware designed for robotic systems, that manages communication between the RGB-D camera and the DoBot Magician. ROS's publish-subscribe model allowed different software nodes to operate autonomously yet collaboratively within a distributed network (Open Robotics, 2020). It enabled real-time data transfer and controlled the DoBot's movements precisely.

2.2.2.1 RViz

This is a 3D visualisation tool in ROS that allows users to view data from sensors and simulations in real-time (Hershberger et al., 2019). It is used for visualising camera data: displaying point clouds, depth images, and object detection and the software also aids in debugging.

2.2.3. Linux

Linux is an operating system specificly used in the Ubuntu 20.04 LTS distribution (Gurjeet, 2020). This serves as a primary operating system to run ROS and camera and robot interface. Linux's compatibility with ROS (Noetic) made it the preferred platform for this project, ensuring smooth installation and running of ROS packages and dependencies necessary for the RealSense camera and DoBot integration.

2.2.3.1 Windows Subsystem for Linux (WSL) / CLI

Provided a Linux-compatible environment, WSL enabling the team to run ROS and interact with the camera directly from Windows laptops. Key functions included: the Linux Command Line Interface and Compatibility with MATLAB. WSL provides access to linux commands, allowing control of ROS, launch nodes, and access the RealSense camera data. WSL also bridges the gap between Windows-based MATLAB and the Linux-based ROS, creating seamless communication and ROS bag recording for MATLAB analysis (Harisudhan.S, 2024).

2.2.4 Github/Github Desktop

Github was utilised for source control. All group members had access to a common repository for uploading code as it was developed.

3.0 Methodology and User Guide for Code

This section runs through the project step-by-step in terms of set-up, configuration of code, object detection, picking, placing and sorting using the DoBot and RealSense camera. This section provides clarity on how the project was created and designed, with each step focusing on allowing easy replication of the project if wishing to do so.

3.1 Installation

The majority of this section will redirect you to other files provided by subject 41014 to assist in downloading the required software and toolboxes. If completing/ed course 41014 this step should already be completed:

1. MATLAB: Follow the Installation of MATLAB and Simulink PDF (MathWorks, n.d.) for installation. Be sure to have the following toolboxes downloaded; **MATLAB,** Simulink, **Optimisation Toolbox**, **Robotics Systems Toolbox**, **ROS Toolbox,** Simulink 3D Animation, Simulink Real-Time, **Image Processing Toolbox**, **Computer Vision Toolbox**, **Statistics and Machine Learning Toolbox,** and the Symbolic Math Toolbox. [*Note: The **bolded** toolboxes are key components to completing this project*]
2. Ubuntu OS: This is required to download ROS, as it only works with Linx-based systems. Firstly Install Ubuntu desktop | Ubuntu then download Ubuntu 20.04 LTS at Ubuntu 20.04.6 LTS (Focal Fossa)
3. ROS: The specific one which is needed to be installed is ROS Noetic Installation on Ubuntu
4. Intel RealSense: Through the canvas portal through the following link.
5. MATLAB-ROS Integration: Configure MATLAB to communicate with ROS, allowing for communication between the RealSense camera and DoBot.

3.2 Running Code

The team developed a script that allows for communication between all hardware and software.

The initial setup involves:

1. Starting ROS
2. Launching RealSense Camera Node and Recording a ROS Bag
3. Running Main Script in MATLAB: This includes loading the ROS bag, performing colour and shape detection, and picking and placing counters in defined spots.

***Note**: in the case an object isn't detected, the system prompts an error and requests a re-recording of the ROS bag.*

```
function test_opt()
% Clear workspace and close figures
% clear all
% close all

% Step 1: Start a persistent WSL session by launching a basic command
disp('Starting a persistent WSL session...');
system('wsl -d Ubuntu --exec /bin/bash -c "sleep 300" &');  % Keeps WSL active for 5 minutes

% Step 2: List USB devices in Windows and prompt user for the bus ID
[status, usbList] = system('usbipd list');  % List USB devices
disp('Available USB Devices:');
disp(usbList);

% Prompt user to enter the bus ID
busID = input('Enter the BusID of the camera to connect to WSL (e.g., "1-6"): ', 's');

% Step 3: Bind and attach the selected USB device to WSL
bindCommand = sprintf('usbipd bind --busid %s', busID);
[status, bindOutput] = system(bindCommand);
disp(bindOutput);

attachCommand = sprintf('usbipd attach --wsl --busid %s', busID);
[status, attachOutput] = system(attachCommand);
disp(attachOutput);

% Step 4: Verify USB connection in WSL by showing `lsusb` output
disp('Verifying USB connection in WSL...');
[status, lsusbOutput] = system('wsl lsusb');
disp('Connected USB devices in WSL:');
disp(lsusbOutput);

% Step 5: Launch the camera
wslCommand = "bash -c ""source /opt/ros/noetic/setup.bash && source " + ...
    "~/catkin_ws/devel/setup.bash && roslaunch realsense2_camera rs_d435_camera_with_model.launch""";
% wslCommand = "bash -c ""source /opt/ros/noetic/setup.bash && source
%  ~/catkin_ws/devel/setup.bash && roslaunch realsense2_camera rs_camera.launch""";
system("start wsl " + wslCommand);

% Pause to allow the camera to initialize
pause(10); % Adjust as needed
end
```

3.3 Developed Code and Research

This section covers the knowledge acquired and/or required to create Project 3 in the way defined in Section 1.2. [*Note: point clouds were a considered approach, and provided another perspective on how to approach this project, it is given in section 3.4*]

3.3.1 Object Detection

Object detection is achieved through the use of the Intel RealSense RGB-D camera (2.1.2), which allows us to record both the RGB and depth data topics from the camera for image processing. MATLAB (2.2.1) handles the image processing, using colour and shape analysis to identify objects based on their characteristics.

How to do object detection:

1. Colour: Using MATLAB's *rgb2hsv* function converts the RBG images to HSV spectrum, which enables us to much more easily mask out the red, blue and green in the captured images.

```
% Convert RGB image to HSV for red masking
hsvImg = rgb2hsv(img);

% Define thresholds for red hue, saturation, and brightness
hueLow = 0.0;
hueHigh = 0.05;
hueHigh2 = 1.0;
hueLow2 = 0.95;
satLow = 0.5;
valLow = 0.2;

% Create a binary mask for red regions
mask1 = (hsvImg(:,:,1) >= hueLow & hsvImg(:,:,1) <= hueHigh);
mask2 = (hsvImg(:,:,1) >= hueLow2 & hsvImg(:,:,1) <= hueHigh2);
mask = (mask1 | mask2) & (hsvImg(:,:,2) >= satLow) & (hsvImg(:,:,3) >= valLow);

% Apply the mask to the depth image
masked_depth = d_img;
masked_depth(~mask) = NaN; % Set non-red areas to NaN to ignore them
```

2. Detects Largest object: The object of the largest size within the masked image is selected.

```
% Find the centroid of the top face region
top_stats = regionprops(top_face_mask, 'Area', 'Centroid');
if isempty(top_stats)
    error('No top region detected');
end

% Select the largest connected component as the top face
[~, idx] = max([top_stats.Area]);
center = top_stats(idx).Centroid;
```

3. Centroid: The centroid of this largest shape is found and the coordinates are marked in the image.

```
54      % Convert the centroid to integer pixel values
55      x = round(center(1));
56      y = round(center(2));
57      depth = d_img(y, x); %ISSUE HERE --- NAN or 0 error
```

4. Conversion to 3D: The centre point of the detected object is then converted to a 3D point in the camera frame.

```
% RealSense D435i camera intrinsic parameters
fx = 611.82763671875;
fy = 611.438232421875; %
cx = 323.9910583496094;
cy =  232.9442901611328;

% Calculate 3D coordinates in the camera frame - from quiz 1
X = (x - cx) * depth / fx;
Y = (y - cy) * depth / fy;
Z = depth;

```

5. Translation and Rotation: Matrices are then applied to convert the 3D point to the Dobots frame, ready for pickup.

3.3.2 Grasp Point Calculation

In regards to the 3D grasping points they can be calculated by combining RBG data with depth information from the realSense camera. This depth data gives the Z-axis positioning. More specifically, The centroid's X and Y coordinates from the RGB are combined with the Z-axis to define a precise 3D point.

For our initial development however we ignored the Z value as we knew the height of the objects. This helped speed up development speed.

3.3.3 Robot Control

The DoBot is controlled using ROS and a provided class from Industrial Robotics that handles the basic ROS publishing and subscribing and a developed function. The developed function takes two points (a pickup and drop off).

Inverse Kinematics calculations by us are NOT REQUIRED as the Dobot can take a desired point for the end effector and can conduct inverse kinematics, compute Q angles and generate a Q-Matrix internally. It is important to note that it appears to take a linear path from point A to B, thus waypoints were implemented to avoid potential collisions.

```
function moveFromP1toP2(P1,P2)
%A function for moving a Dobot magician to a specific location, picking up
%and then dropping off.
    dobot = DobotMagicianRealMove(); %create the Dobot object that handles ROS pub and sub

    rot = [0,0,0]; %rotation matrix for rotation of end effector (ignore if using suction cup);

    dobot.PublishEndEffectorPose([P1(1),P1(2),P1(3)+0.1],rot);
    pause(3); %pauses needed as the publish is not a blocking function during the movement.
    dobot.PublishEndEffectorPose(P1,rot);
    pause(3);

    %Turn suction gripper on to pickup object
    onOff = 1;
    openClose = 1;
    dobot.PublishToolState(onOff,openClose);
    pause(0.3);

    %move up to avoid collisons
    dobot.PublishEndEffectorPose([P1(1),P1(2),P1(3)+0.1],rot);
    pause(3);

    %move ot drop off
    dobot.PublishEndEffectorPose([P2(1),P2(2),P2(3)+0.1],rot);
    pause(3);
    dobot.PublishEndEffectorPose(P2,rot);
    pause(3);

    %Turn suction gripper off to drop off object
    onOff = 0;
    openClose = 0;
    dobot.PublishToolState(onOff,openClose);
    pause(0.3);

    %move up to avoid collisons
    dobot.PublishEndEffectorPose([P2(1),P2(2),P2(3)+0.1],rot);


end
```

1. Points for the pickup are generated by the camera and object detection algorithm (passed to the function as P1) and the drop-off is predetermined (passed to the function as P2).

2. Control Sequence: It moves the counter, picks the counter using the suction cup, moves to the arrangement location and then releases the counter - This is repeated until the desired arrangement is completed.

4.0 Results and Testing

For testing, we utilised the Dobots [0,0,0] as the start of the global coordinate from. For the main method for working on and improving the system was as follows

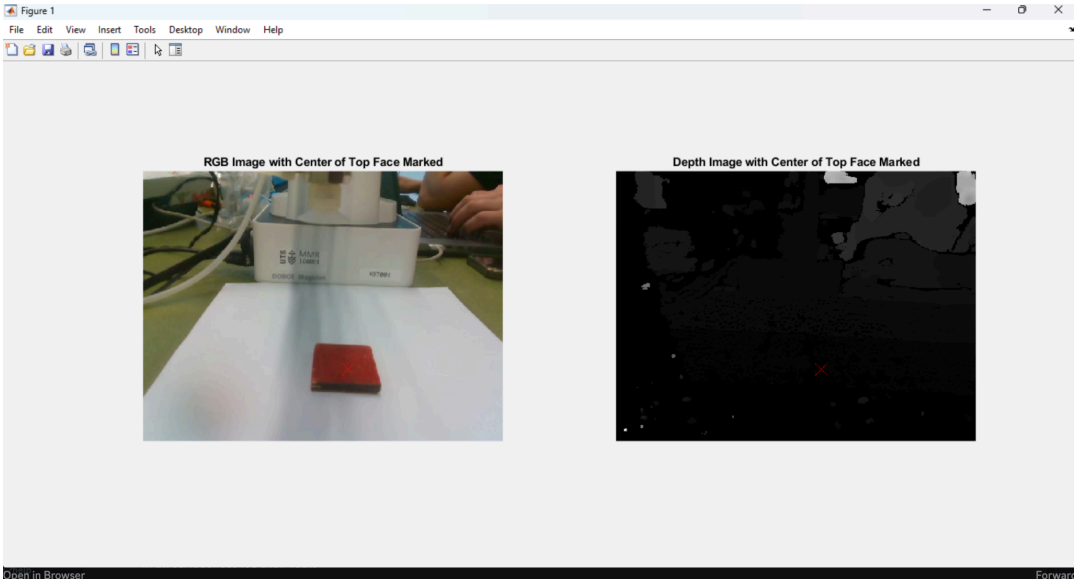
1) Place the object in a known position in the global coordinate frame.
2) Capture an image/ROS BAG
3) Pass the bag through our algorithm and generate a proposed point of the object
4) Move the Dobot to the generated positon.
5) Check how close it is

From these test some of the factors we would adjust are:

1) Camera X Offset- A translation from the end effector to the eye in hand camera
2) Camera Y Offset - A translation from the end effector to the eye in hand camera
3) Camera Z Offset - A translation from the end effector to the eye in hand camera
4) The relationship between the camera/axis/frame of reference and global axis

4.1 Object Detection

Our object detection algorithm was able to detect the circles in blue and red very consistently and Green somewhat consistently. Below shows the initial testing of this algorithm (before switching to eye in hand).



4.2 Pick and Place

The Dobot was extremely consistent with pick and placing as long as the point was accurate and towards the middle of the object. If it was not, the suction gripper could not create a vacuum to pick it up.

4.3 Performance

Overall the performance of the system was progressing towards what we had hoped. While development was affected due to the reasons outlined below we were happy with how it was progressing and believe the system could be further developed to perform more complex tasks.

4.4 Challenges and Solutions

Challenge	**Details of Challenge and Solutions**[1]
Overall project related	
Project Selection	Originally when selecting the project the team consisted of 3 members with only one not being able to use all the software required for the original project. When the additional member joined, we reassessed the project and identified that 2 individuals would not be able to fully participate in the project - thus resulting in the change of the project to Project 3. Even with the warning of limited resources. The team managed to come in early to prevent this from being a large problem.
Learning Curve	A couple of the concepts and software were relatively new. Specifically aspects of ROS and RealSense integration, this caused our progress to go slower. To overcome this, we did research on ROS and RealSense systems to better understand how they work and their capabilities.
Time Management	There were a couple of issues out of our control such as hardware issues and complex integration, slowing our progress down. Additionally, aligning everyone's schedules to ensure at least one member who had access to ROS was able to be present to work on the project was challenging, due to our team's busy schedules. Only having 2 weeks to work on the project and troubleshoot was a bit challenging however in our project scope we did have an outline of how and when we would aim to achieve each component. Possibly more time or a stronger timeline, with the ability for flexibility would have helped in ensuring the project was achieved within our preferred deadline. Moreover, the time restrictions we had for working with the equipment prevented us from making major breakthroughs with the hardware. However, being provided a camera allowed us to trouble shoot out of hours and have something to implement to the project when all hardware was set up.
Hardware Related	
Raspberry Pi Functionality	Multiple Raspberry Pi devices were tested (almost every one the lab had), and were inoperable as their systems were being updated. There were considerations of purchasing our own Raspberry Pi - however, the time it would have taken to upload all firmware would have taken time we didn't have as well as the cost of the specific Pi required was quite pricey, and would have been challenging to share after the project was completed. Issues were brought up with tutors and lab technicians, they were solved in the next couple of days - First having a list of the working Raspberry Pi's and over the weekend assigned them through to each DoBot, thus making connectivity easier and more achievable.

[1] *Note: solutions are either fixes we found online and or came up with*

	This issue caused a SIGNIFICANT delay in development for the whole group.
Camera Placement and Calibration	It was a consistent challenge to ensure the camera was positioned in the same spot each time in which the program was operating. As a result, we switched to an eye-in-hand setup by attaching the camera to the end effector pointing straight down.
Software Related	
MAC OS Compatibility	Half of the group was utilising Macbook laptops. As a result, they can not run Linux, WSL and therefore can not use the intel realsense cameras. This made development extremely difficult for those students and as a result, they were limited to only working on the Dobot. Additionally, as mentioned in the time management section as only 2 members had access, it required one of them to be present when working
General Coding	There is alway going to be challenges with coding, many times we came up with errors. To overcome theses errors, it mainly involved a lot of team work to refine and resolve the issue, plus additional research.
Inconsistent Translations	The transformations between the camera's data feed and the DoBot's coordinates created errors in localisation and objection picking and placing. Overcoming this involved a decent amount of guess and checking plus adjustment of the translation matrices in MATLAB and ROS. Consistent alignment of test runs and having multiple ROS bags allowed us to have more data for more points thus making it easier to fix theses inconsistencies.

5.0 Future Works

There are a couple of areas in which this project could be improved to enhance its usability both technically and in the OT world. This section will cover each relevant section and how it can be improved.

5.1. Detection and classification

To improve the accuracy and adaptability of the object detection system - the following can be implemented:

- Although attempted this is more so a future implementation to enhance the current system. The use of **Point Cloud processes** can enable precise detection and classification of objects with varying shapes, sizes, and textures. This makes the system more engaging and challenging for clients.
- **Machine Learning Algorithms for Classification:** The potential of making the system adaptable to different characteristics, using a larger dataset of color, shape, and size allowing the project to go beyond simple geometric shapes. And be used to teach life skills, like setting a table.

5.2 User Interface (UI)

In order to better serve the needs of OT's a couple adjustments can be implemented for more personalised lessons. Additionally it would expand the picking and placing capabilities.

- **Advanced UI for customisable arangements**: It would allow for therapists to define and save custom arrangements based on specific client needs. This further woudl include real-time arrangement selection and configuration adjustments, such as modifying object positions, colors, or sequence orders. (would require section 5.1)
- **Predefined Object Arrangements**: Creating a predefined arrangements (e.g. arrangement 1 = red, blue, green in sequence) can be stacked or next to eachother, and include the same or different shapes. This could streamline the setup process, allowing therapists to select from standard options or create new configurations based on their session goals. Each arrangement could be hardcoded with specific parameters for colour, shapes and positions providing versatility.

5.3 System Integration and Safety Enhancements

As this project is to be used in a clinical settings, the system's usability and safety features would have to be improved to ensure Work, Health and Saftey regulations are complied with:

- **Real-Time Adaptability**: This allows the therapists have a stop button along with the ability to make adjustments mid-task, increasing the complexity of the lesson, and safety.
- **Collision Detection**: Additional sensor to ensure the robot can stop in the case the clint gets to close to the DoBot.
- **Error Recovery Protocols**: This would enhance reliability, reducing the need for manual intervention and increasing the system's resilience in continuous use.

5.4 Data Collection for Therapy Progress

To make the system more effective as a therapeutic tool, advanced control features and data analytics capabilities could be incorporated:

- **Detailed Motion Control and Data Storage:** Having the robot record each movement path, along with task completion time and error rates, would enable therapists to track client progress over time. To further advance this idea would be to include **Other Sensory Systems**. This helps to track clients interactions with objects, such as hand movements or grip strength, providing therapists with more comprehensive feedback, to create a more tailored program.
- **Data-Driven Customisation**: Based on the way in which the client is reacting to the tasks and the system could recommend specific exercises or arrangements to develop their skills further.

6.0 Conclusion

In summary this project successfully demonstrates the integration of the DoBot Magician robotic arm with the Intel RealSense RGB-D camera for object detection, with picking and placing, in the defined space - meeting the project scope of picking and placing. Through the use of MATLAB and ROS, the project was able to achieve real-time processing and DoBot movements, allowing the robot to pick and place the counters.

Further more the project accomplished effective: object detection and grasping - through the use of colour segmentation and centroid detection; Robot controls - enabling smooth and controlled movements, with the robot reliability executing pick and place operations; and real world application of OT lesson implementation to develop fine motor skill and colour recognition. However, while the project achieved its primary objectives, challanges with hardware reliability, time constraints, and callibration issues created obstacles. Nevertheless theses obstacles prov ided valuable insights to the project limitations and areas for improvement in future designs.

All in all the project demonstrates the potential for developing accessible, customisable, and safe robotic systems for therapeutic use, particular in the OT sector. With further refinements in the system this project can provide therapists with a tool that is both versatile and effective.

7.0 References

Ademovic, A. (n.d.). *An Introduction to Robot Operating System: The Ultimate Robot Application Framework*. Toptal Engineering Blog. https://www.toptal.com/robotics/introduction-to-robot-operating-system

DOBOT Magician. (n.d.). *DOBOT Magician | An all-in-one STEAM Education Platform*. Www.dobot-Robots.com. https://www.dobot-robots.com/products/education/magician.html

Gapaul. (n.d.). dobot_magician_driver/matlab/DobotMagician.m at master · gapaul/dobot_magician_driver. GitHub. https://github.com/gapaul/dobot_magician_driver/blob/master/matlab/DobotMagician.m

Gurjeet. (2020). UBUNTU LTS SERVER OPERATING SYSTEMS. In International Research Journal of Modernization in Engineering Technology and Science (pp. 2582–5208). https://www.irjmets.com/uploadedfiles/paper/volume2/issue_11_november_2020/4900/1628083195.pdf

Harisudhan.S. (2024). Windows Subsystem for Linux (WSL). Medium.com. https://medium.com/@speaktoharisudhan/windows-subsystem-for-linux-wsl-f7e0ba304159

Hershberger, D., Gossow, D., & Faust, J. (2019, April 27). rviz Documentation. https://docs.ros.org/en/lunar/api/rviz/html/c++/index.html#:~:text=rviz%20is%20a%203d%20visualization,capabilities%20in%20your%20own%20applications.

Intel®. (2024). *Intel® RealSense™ Depth Camera D435*. Intelrealsense.com. https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435.html?srsltid=AfmBOooCgzTqkXIGCC4RX2lelUc3Q2xSoPnpeK0__MhsfYr7v07iE8DO

MathWorks. (n.d.). *Installation of MATLAB and Simulink Contact: academicsupport@mathworks.com*. Retrieved November 7, 2024, from https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/academia/student-competitions/best-robotics/files/Installation_of_MATLAB_and_Simulink.pdf

Open Robotics. (2020). *ROS.org | Powering the world's robots*. Ros.org. https://www.ros.org/

Yuejiang, S. (2019). *User Guide Dobot Magician User Guide*. https://wiki.idiot.io/_media/dobot-magician-user-guide-v1.7.0.pdf