Study Oriented Project

on

Robotic Small Object Detection and Pickup

End Semester Report Submitted By

Akshaya Venugopal (2021AAPS0022P)

Under the guidance of

Dr. Avinash Gautam

(CSIS Dept, BITS Pilani)

Table of Contents

1. Introduction

The ViperX-300 6 DOF Robot Arm, part of the Interbotix X-Series, by Trossen Robotics features advanced DYNAMIXEL X-Series Actuators for enhanced torque and compact design. It has two kinds of servos, DYNAMIXEL XM540-W270 & DYNAMIXEL XM430-W350 servos, and offers precise control. The Robot's DYNAMIXEL U2D2 enables easy interaction through DYNAMIXEL Wizard software and ROS.

Fig1: ViperX-300 6 DOF Robotic Arm

This project involves the usage of the ViperX 300 robotic arm to detect, pick and place objects in a given workspace. Using an overhead camera, and later a camera mounted on the robotic arm and different methods of object detection, we can use this arm to perform a wide range of tasks, ranging from object sorting to human assistance.

The main software used is Python, OpenCV and ROS. In terms of hardware, apart from the arm, the setup involves an overhead camera and an Intel Realsense Camera, mounted on the arm for real time object detection from the robot itself in the absence of the overhead camera.

2. Weekly Progress

Week 1:

- I went through the Trossen Robotics documentation about the Robotic Arm, and installed the various dependencies for simulating the arm on my system.
- I went through the Python API and the various examples, changing around different commands and testing various responses on the Rviz Simulation. I learnt about the various types of functions for moving the arm to a specific point.
- I was thorough in going through the modules on github, understanding what values they take in and their breakdown, so that I would be able to use them without any problem in the coming weeks.
- I wrote some basic Python codes to test on the simulation to understand the commands better.

Week 2:

- This week I spent time working on the pose estimation with an overhead camera.
- I used a code on object tracking to understand how this works. I then recreated the code, and modified it to just detect the object position in pixels.
- The next step was to normalise an origin and find the position of the object in cm with respect to the origin that I had specified.
- The next step was to try and figure out how to feed the coordinates to the arm and get the arm to reach the right coordinates.
- For this I tried multiple tests, and finally did the math for how this conversion would work.
- I took the x and y values from the camera pose code in cm, fed the x directly to the arm with an arbitrary z value, as that doesn't concern the movement, as long as I move the arm the same amount in the negative direction after the motion. Then calculate the inverse tan value of (y/x) to get the value of the angle needed for the rotation of the arm joint. This angle will be fed to the arm and hence move it in the required y direction.

Week 3:

This week I worked on the finishing touches to the pick and place implementation.

- I tested the rotation motion for reaching the Y coordinate, and verified it in simulation
- I have also modified the x coordinate that is being fed into the arm, by feeding the square root of the X and Y values first into the arm, as that makes more sense for the rotation.
- I am yet to test this on the arm directly and calibrate the camera for the arm's specific workspace.
- I started exploring the Visual Servoing Platform and just trying some basic tutorials.

Week 4:

- This week was mostly spent with setting up and troubleshooting the arm connection with the NuC.
- This took longer than expected due to the issue that the arm was not being detected by the Nuc and it was only being detected by the VM on my laptop.
- After trying many different methods, I finally realised the issue was coming from the type of USB connector that was being used.
- The disconnected state of the arm had the U2D2 microcontroller only showing a red light, and in the connected state the blue, green and red lights were all on.

Fig 2 : Image of successfully connected arm

Week 5&6:

- These two weeks involved shifting the setup onto the NuC completely, and running the code from there instead of the VM
- This week also involved the camera calibration. One of the biggest issues with the first camera I was using was that it was highly unstable on the mount. This week I tried a different camera which was relatively more easy to use and attached firmly to the mount.
- This week the arm was finally successfully able to pick up the object. There were some issues with the movement of the mount, but this will be resolved in upcoming weeks.
- I also started on the tracking of the object code this week.

Week 7-8(Pre-Midsem to Midsem Week):

- This week I implemented the colour based pick and place, by modifying the code of basic pick up and place (the detailed breakdown of which is in the code breakdown section of the report).
- Upon sir's suggestion, I have started to set up the MOSSE tracker for the issue that I am currently facing with the origin.
- I am also working on tweaking the tracking code that I started pre Midsem.
- Alongside this I am also working on the longest edge detection that was suggested after the coloured pick and place demo.

Week 9 (Oasis Week):

- This week I started work on learning how to use the MOSSE Tracker and learning how to use it for fixing the issue of the moving origin.
- This first involved learning about Aruco Markers and implementing a detection code for the Aruco Marker that would be placed on the table.
- Through this, I learnt that Aruco marker detection would be more reliable to use than the detection through colour as I had done prior for the pick and place part that I had implemented after mid semester.

- The next step was learning about different trackers like the MOSSE tracker and how they worked. The further implementation of this happened in the subsequent week.

Week 10:

- There was a slight setback this week as I had fallen ill after oasis and progress had slowed down.
- I implemented a basic MOSSE tracker, that would track a specific square in a camera frame, which could be selected by the user from a still of the frame.
- The next stage was putting this together with the Aruco detection and tracking code from the previous week.
- There were some issues in this step as there Aruco tag was being detected when selected in the frame, but was not feeding the correct coordinates to the main code.
- There turned out to be a difference in normalisation of the coordinates in the two different trackers. Once this was corrected, the tracker worked perfectly fine and now the system was set up in such a way that even motion of the camera would not impact the working of the whole setup.

Week 11:

- This week I worked on using the Intel Realsense Sensor for object detection.
- The first issue was of setup of the sensor on the arm, we had to figure out a way to mount it on the arm.
- The current way is that it is mounted with tape and some rubber bands to secure it on the arm, however it is difficult to move and falls sometimes.
- This setup is temporary, moving forward we will try to find a more stable setup.
- Then I worked with some of the example codes given for the sensor, and used it to calculate the position of an object placed a distance from the sensor.
- As well as position, using an MobileNet, an object detection model I was able to detect the type of object as well.

- There was an issue of lighting as the object wasn't clearly seen by the sensor, as there was a lack of light in the area, which we will aim to get fixed moving forward.

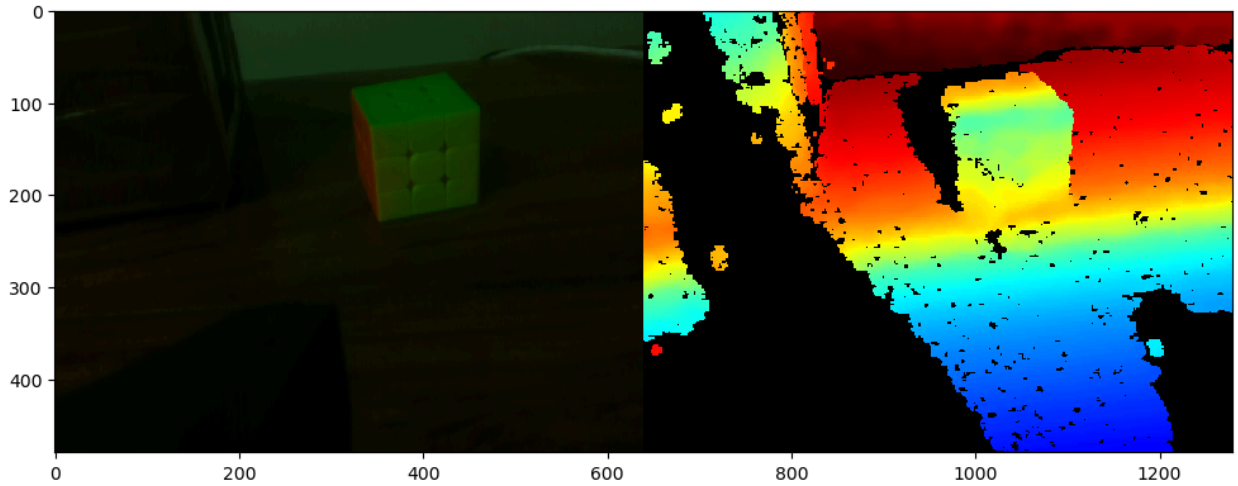

Figure 3: The image captured by the sensor and its corresponding depth data

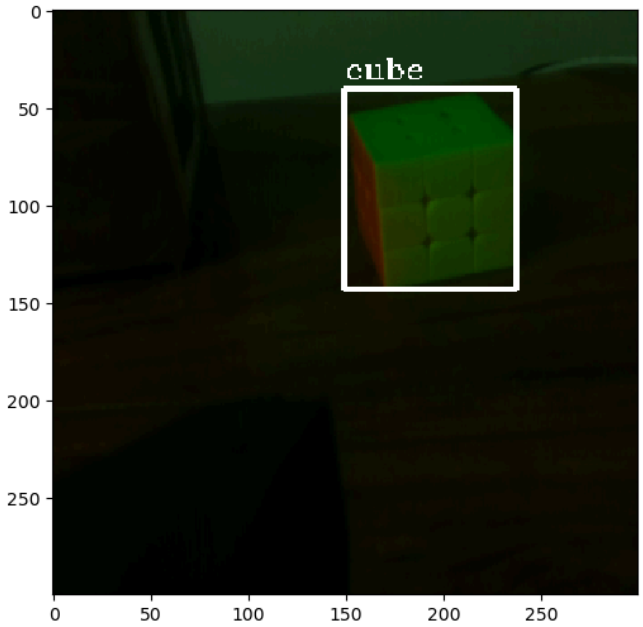

Figure 4: The result of the object detection on the image frame

3. Technical Details

-All of the code for this project was written in Python, and done on an intel NuC system with Ubuntu 20.04 and ROS Noetic.

-The Python libraries used for the same are:

- Interbotix_xs_modules.arm - The arm SDK which allows us to work with the robot
- Math - to do some basic operations like tan calculations for coordinates being fed to the arm
- Imutils - to access the webcam/overhead camera
- OpenCV (cv2) - for computer vision
- Matplotlib.pyplot - to plot the images gotten from the Realsense Sensor
- Pyrealsense2 - the SDK for the realsense sensor
- Numpy - for matrix/image operations
- Time - for execution speed calculations and for suggested speed of the robotic arm.

- Explanation of the various codes and their purposes in my project folder:

- Camera_tracking_test.py - This is a basic code that just finds an object of a specified colour in the frame and prints the coordinates of the geometric centre of the object. This is then converted to cm, from its original pixel value. This was modified from a pyimage search tutorial on basic object tracking. In the updated version, it finds the origin with the find_origin function then finds the coordinates of an object specified by colour wrt to the origin.
- Arm movement.py - Has two different functions defined for the robotic arm, one for pick up and another for drop, that take in the coordinates of the object in the workspace as input and the arm moves to the required position and picks/drops the object based on the function called.
- Find_obj.py - Does the basic function of picking up an object with a fixed colour, which can be specified by the user.
- Pickup_and_drop.py - takes user input of colour of object to pick up, and colour of where to drop it off and the arm will take the required object and

drop it in the specified space.

- Tracking_object.py - will track the position of the object and if the object remains in the same position for 2 or more loop iterations, the arm will pick up the object.
- Aruco_detect.py - The code detects an Aruco code of a specified type and dimensions in the camera field of view, and marks its coordinates in the frame and shows the coordinates of the Aruco tag on the frame
- Aruco_type.py - Finds the type of the Aruco Tag used in an image.
- Distance.py - A pick and place application that doesn't use the dynamic origin. (Pre-Midsem implementation of the pickup code)
- Origin_finder.py - This code combines a MOSSE Tracker with an Aruco detection code, to detect and track the exact coordinates of an Aruco Marker in a frame. This helped to resolve the issue with the moving origin. As the marker is stationary wrt the origin, we can use this to calculate the origin in any situation.
- colour_find.py- a code to find the threshold values in HSV for any specific colour. This was taken from a tutorial (referenced in the list of references section of the report)
- Object_tracker.py - Implements a simple MOSSE tracker in which a user can select a segment of a frame to be tracked (followed a tutorial from pyimagesearch for the same - referenced in the end of the report)
- Distance_to_object.ipynb - This uses the RealSense Sensor and finds the distance to an object detected by the sensor.
- Realsense_record.py - A code to test the recording capability of the realsense sensor.
- Realsense_test.py - A basic code to test the functions of the RealSense Sensor.

4. Problems Faced

-The arm would initially not connect to the NuC, but later this was resolved, as the issue was just due to the fact that the usb cable of the arm was USB 2, and was unable to be recognized by the NuC's ports that are USB 3

-The motion of the camera is causing issues to the normalisation of the origin for every time the program is run - the suggestion was made to use the MOSSE tracker for this, and I am currently testing it.(This was resolved Post Midsem)

-The arm is unable to pick up an object if it is rotated such that the biggest side is in the direction of the arm pick up. This was specifically evident in the rubik's cube case. This needs to be fixed, I have tried to use a method of finding the length of the longest contour, but it isn't perfect yet.

- The colour detection method for picking up objects isn't very reliable, as the colours look different in the camera in different lighting conditions giving erroneous values sometimes. This can be resolved using Aruco markers for pick and place.

- The MOSSE Tracker and the Aruco detection when put together were feeding conflicting values for the origin. This was later resolved, as I realised that the MOSSE Tracker and the Aruco detection were taking different frame sizes, hence having different origin values.

- The RealSense sensor is having a few issues namely:

- The current setup of the RealSense sensor involves it being taped to the robot and secured with rubber bands, but this isn't very secure as it is falling and facing the ground. A possible solution will be to attach this to a mount and mount it on the arm with screws.
- There is a lack of lighting in the area where the sensor is capturing images so it is not getting very defined images, and unable to recognize any objects in the frame.

5. Future Goals

- Make sure the object can pick up any rotated object, by finding the smallest side and pick up the object

- Ideally, remove the overhead camera and only use the Realsense Sensor for any object detection with the arm.

- Make the robot capable of sorting objects, which have the same colour or tag

- Make the tracking more smooth and perfect. The current implementation is very rudimentary and needs further improvement.

- Incorporate a voice command system for the code, to make it more interactive for any user.

- Migrate from python and open CV to using the Visual Servoing Platform (ViSP), as this way we can use pre existing codes and make further improvements faster

6. Acknowledgements

I would like to thank Dr. Avinash Gautam sir for giving me this project and always guiding me throughout the semester. I would also like to thank sir for giving me ideas to solve any issues I faced at any point during the semester. When I first took up the project I felt that I didn't have enough qualifications for this project, but I am proud of what I have managed to do in this semester.

I would also like to thank the PhD research scholars of the INSPIRE lab as they were always there to help me with any requirements I had for the project, as well as guide me in case I was having an issue with implementing anything or with any problems with installations and code.

It was a pleasure to work in the lab this semester and I look forward to continuing to work in the lab in the upcoming semester as well.

7. List of References:

1. Rosebrock, A. (2021, April 17). *Ball tracking with opencv*. PyImageSearch. https://pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/
2. automaticaddison, A. (2020, September 6). *How to convert camera pixels to real-world coordinates*. Automatic Addison. https://automaticaddison.com/how-to-convert-camera-pixels-to-real-world-coordinates/
3. *Find and Draw Contours using OpenCV | Python - GeeksforGeeks*. (2019, April 29). GeeksforGeeks. https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/
4. Praveen. (2020, July 28). *How to find HSV range of an Object for Computer Vision applications?* Programming_fever. https://medium.com/programming-fever/how-to-find-hsv-range-of-an-object-for-computer-vision-applications-254a8eb039fc
5. Rosebrock, A. (2021, April 17). *Detecting ArUco markers with OpenCV and Python - PyImageSearch*. PyImageSearch. https://pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-and-python/
6. Rosebrock, A. (2023, June 8). *OpenCV Object Tracking - PyImageSearch*. PyImageSearch. https://pyimagesearch.com/2018/07/30/opencv-object-tracking/
7. IntelRealSense. (n.d.). *librealsense/doc/distribution_linux.md at master · IntelRealSense/librealsense*. GitHub. https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution_linux.md

8. Rosebrock, A. (2021b, April 17). *Live video streaming over network with OpenCV and ImageZMQ - PyImageSearch*. PyImageSearch. https://pyimagesearch.com/2019/04/15/live-video-streaming-over-network-with-opencv-and-imagezmq/

A personal Notion Page I have created for all links related to this project : https://highfalutin-dirigible-13a.notion.site/Robotic-Small-Object-Pick-and-Place-4a598fd4e8ed4865accec02705514757?pvs=4