Laboratory Oriented Project

on

Unseen Object Segmentation for Robot Grasping

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. This project of object segmentation aims to improve on the work done in the previous semester on object detection and pick up.

Fig1: ViperX-300 6 DOF Robotic Arm

In the previous semester, the ViperX 300 robotic arm was used to detect, pick and place objects in a given workspace. At that time, the object was detected using an overhead camera and OpenCV for colour thresholding to detect the object. But this method had some downsides and wasn't applicable to the purpose of object tracking. In this semester, I tried to incorporate the use of modern techniques in object tracking and detection to smoothen the process of object tracking in real time.

The main software used in the previous semester is Python, OpenCV and ROS. Additionally, this semester I attempted to use ML models like XMem and MMFlow for object tracking and optical flow estimation. In terms of hardware, apart from the arm, the setup primarily involves an overhead camera.

2. Weekly Progress

Week 1:

- The first week primarily focused on the set up of the lab system and modifying of the code for the demo for the students who were visiting the lab as part of the Young Entrepreneurs Bootcamp (YEB)
- This setup involved modifying the code from the previous semester to pick up the objects from specified locations on the table, and dropping the objects to other locations on the table.
- Other than this, I also started on the readings of the papers 'Self-Supervised Unseen Object Instance Segmentation via Long-Term Robot Interaction" and "EasyLabel: A Semi-Automatic Pixel-wise Object Annotation Tool for Creating Robotic RGB-D Datasets"

Week 2:

- I continued my reading of the papers and created a presentation based on my understanding of the same.
- In brief - EasyLabel is a simple tool to easily acquire high quality ground truth annotation of object at pixel level in densely cluttered scenes, the paper provided a method to create an Object Clutter Indoor Dataset (OCID) with minimum human intervention
- The paper on Unseen object Segmentation involved the use of ML models to improve segmentation of a group of objects placed on a table

Week 3:

- I started working on trying to recreate the setups proposed in the two papers. I started installing the dependencies for EasyLabel and the Unseen Object Segmentation paper.
- For the EasyLabel setup, the dependencies needed involved using Boost, CMake and Point Cloud Library(PCL)
- I also downloaded the pre existing datasets that were made in the EasyLabel paper.

- However due to version dependency issues with PCL, the base code for EasyLabel would not run. I tried to change the version of PCL being installed, but that still didn't work, and the installation of PCL only caused the NuC to crash.

Week 4:

- Since there were issues with PCL I decided to move on to the other paper, the one on Unseen object self-segmentation.
- This paper led me down to read different papers on MSMFormer, which is a Mean Shift Mask transformation and an entirely different project on object segmentation using optical flow estimation and using an ML model for object tracking, called XMem.
- In order to work on the segmentation part, I started to work on the sweeping motion of the robotic arm to push the objects back and forth and improve its segmentation.

Week 5&6:

- While the original plan of this project was to work on a dataset creation with EasyLabel and the ideas presented in the Unseen object supervision to segment and identify a group of objects present on the table, unfortunately due to the issues with EasyLabel, I had to shift the focus towards something that would have a concrete output by the end of the semester.
- I worked on setting up the dependencies as required by the paper on Unseen object segmentation, by downloading the model XMem, and setting up MMFlow.
- While setting up XMem, going through its documentation and demo files and tried to understand how the model works in order to use it for our required purposes.

Week 7-8:

- I had been travelling for a workshop, so I didn't make much progress this week.

- But I did go through the XMem model and figured out that for the model to work, it takes in a first frame and uses a mask of the object to estimate where the object is in subsequent frames, and can track an object quite accurately.
- However there was no codes given on the actual github for the ML model, so now I had to figure out a way to make a code for the mask generation for the first frame

Week 9 (Pre-Midsem Week):

- This week I worked on figuring out a code for the mask generation, which could be given as input to the XMem model
- I tried to reuse some of the code from the previous semester's project, where I had used colour based segmentation techniques to create image masks.
- This did originally have a bit of an issue, as the camera needed time to warm up and sometimes the frame was captured too quickly before the camera had gotten ready.
- In some cases, the mask was finicky and didn't always work, but by the end of this week I was able to get this all set up.

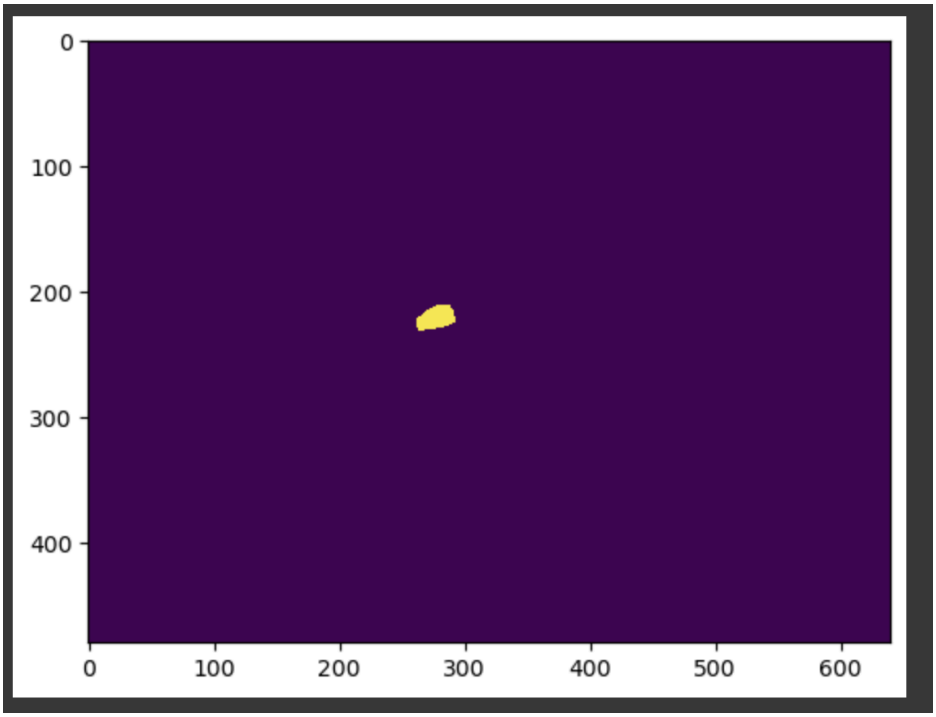

Fig 2 : Mask generated from the first frame

Week 10(Post Misem):

- Now with the mask generated I could use XMem, butI wanted to test it with recorded video.
- I worked on an OpenCV code to record the video and save it but the issue was because I was working on Google Colab, I had to locally record the video and upload it to drive to run XMem on it
- This was turning into a lengthy process, I needed to think of an alternative solution.

Week 11 (APOGEE week):

- I was busy with APOGEE and so I couldn't get much work done. I tried again to set up the video code on Google colab, but this was running into errors each time, so I decided to work on it after APOGEE.

Week 12:

- Based on Avinash sir's suggestion I migrated the code to run it on the server, so that the ML model could run locally instead of on Google Colab
- However this did not solve the problem of the video streaming from the local device. I needed to come up with a solution to send the video from the local camera to the server for processing and send the processed coordinates back to the code that was running on the arm.

Week 13:

- After some thought and different attempts, I decided to use ZeroMQ protocol and the imagezmq python library to send camera frames to the server from the local device.
- This originally didn't work smoothly, but the issue turned out to be in the fact that I was using OpenCV to show the frames on the server side, however this doesn't work on codes running in remote jupyter servers. The alternative method is to use matplotlib to show the images in the jupyter notebook.
- Now the only part that was left is to integrate the code for the coordinates to be fed to the arm.

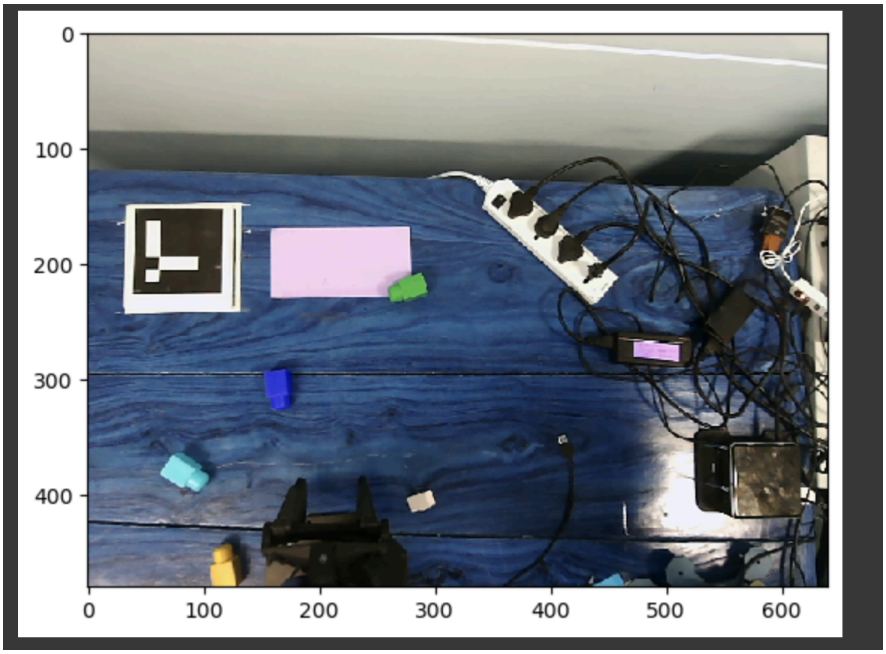

Fig 3: The first frame being sent from the local device to the server(The image is in HSV hence the saturation)

Week 14:

- This week I mainly worked on testing the code for the image streaming, and generating the coordinates from there, using the preexisting code from the previous semester.
- The code was working on the server end, but I needed to find a way to send the coordinates back to the local machine.
- I tried to create a similar ZeroMQ socket and client and server relationship, but it didn't work initially.
- Finally I did figure a way to get it to work, but it worked for one try then the ML model suddenly stopped working, before the demo, unfortunately I can't find a solution for the same at the current moment.

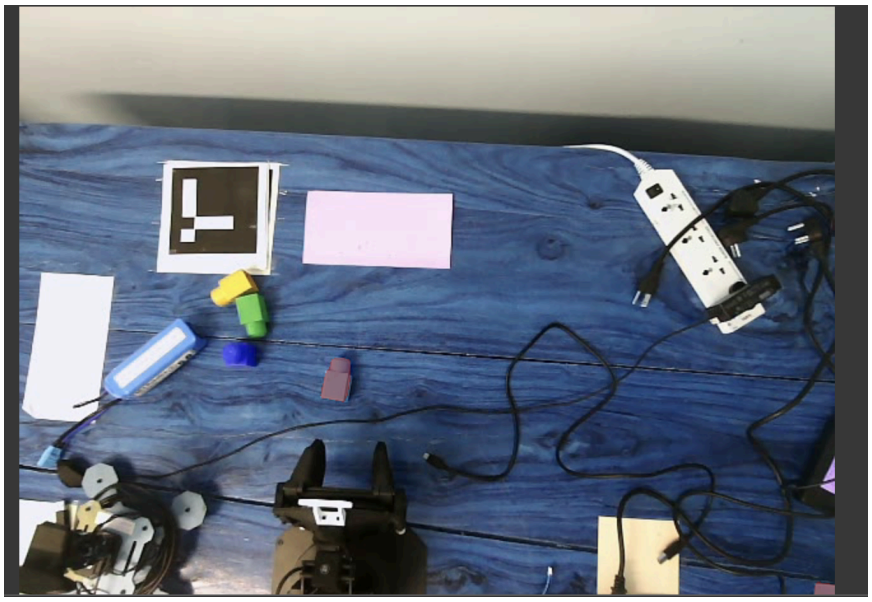

Fig 4 and 5 : Some examples of the Object Tracking, first figure with just the image of the tracking mask on the blue object, and the second figure with the coordinate values being generated on the server side as well.

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. In addition, the ML model is run on the server due the need to use a GPU.

-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 display the images in the jupyter notebook
- Time - for execution speed calculations and for suggested speed of the robotic arm.
- Imagezmq - A library that uses the ZeroMQ protocol for image processing applications
- PIL - An alternative to OpenCV for certain applications to open saved images from a folder
- Pytorch - to set model parameters for XMem and track objects with XMem
- Paramiko - An ssh related python library that helps with sending a file with the coordinate values from the server system to the local system

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

  -XMem_trial_final.py - The jupyter notebook that runs on the server and processes the images that are being sent from the local system. Most of the code is in this specific jupyter notebook

  - client_first_frame.py - The code that runs locally, which sends an image of the first frame to generate the mask of the same.

  -client.py - The code that runs locally to send frames constantly to the server, for constantly processing images

- obj_track.py - The code that takes the coordinates from a text file and feeds them to the arm, for the arm to track the object.

- mask.py - A python code that can be used to generate a mask of a specific object in a frame.

- sweep.py - A python code for the arm to perform a sweeping motion, for object segmentation.

- coordinates.txt - a text file that has the coordinates from the server

4. Problems Faced

- Initially there were a lot of issues with EasyLabel due to the dependency issues with the point cloud library (PCL)
- While there was the goal to recreate the paper on ‘Unseen object segmentation’, due to issues with MMFlow working with XMem, and lack of documentation on how to recreate the whole process, it wasn’t possible to achieve this.
- Originally, there was a lot of struggle to stream the video from the local system to the server for processing, but over time this got resolved.
- In the video streaming setup, there were cases where the Imagehub socket that was being used to send the frame from the local machine to the server was not properly closed. This would cause the server to not launch once again and would require rebooting of the jupyter notebook. Thankfully this was fixed with a bit of code to close the socket, regardless if the code runs into an error or not
- The mask generation ran into similar issues to that of the last sem, when the colour thresholding would not always work, but now it has been resolved
- There is still an issue with the coordinates being sent back to the local machine and unfortunately that has no solution yet.

5. Future Goals

- Before the end of the semester, try to fix the issues with the tracking model, if time permits and have a demonstration for the same.

- 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 or based on text based recognition using ML models, to solve real world problems of sorting objects.

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

6. Acknowledgements

I would like to thank Dr. Avinash Gautam sir for his continued support on this project. Despite the fact that this semester teh progress has been slower due to either issues with the code or because I was busy with other things, he has always been patient with my progress on this project. I will try my best to arrange a demonstration of the work done by the end of the semester as I wish to show all the work that I have managed to finish this semester.

I would also like to thank the PhD research scholars of the INSPIRE lab once again, for helping me with any issues I had and providing me with possible solutions for any troubleshooting.

It was a pleasure to work in the lab this semester as well 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. 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/

8. Hkchengrex. (n.d.). *GitHub - hkchengrex/XMem: [ECCV 2022] XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model*. GitHub. https://github.com/hkchengrex/XMem
9. Lu, Y., Ninad Khargonkar, Xu, Z., Averill, C., Kamalesh Palanisamy, Hang, K., Guo, Y., Ruozzi, N., & Xiang, Y. (2023). Self-Supervised Unseen Object Instance Segmentation via Long-Term Robot Interaction. *ArXiv (Cornell University)*. https://doi.org/10.15607/rss.2023.xix.017
10. Suchi, M., Patten, T., Fischinger, D., & Vincze, M. (2019). EasyLabel: A Semi-Automatic Pixel-wise Object Annotation Tool for Creating Robotic RGB-D Datasets. *ArXiv (Cornell University)*. https://doi.org/10.1109/icra.2019.8793917

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