MRT ASSIGNMENT I

Anushka Verma
22B0068
Aerospace Engineering

February 2023

1 Introduction

ROS (Robot Operating System) is a framework that helps in building software for robots. OpenCV (Open Source Computer Vision Library) is an open-source library that helps in computer vision and image processing tasks.Together, ROS and OpenCV can be used to create complex robotic systems with powerful image processing capabilities.

2 Summary

At first I created my workspace `'catkin_ws'` and added a package called 'camera' into it. I built the package using `'catkin_make'` and started working on the python code for the publisher and subscriber nodes.

2.1 Publisher

The publisher node `'camera_publisher'` publishes to the rostopic `'/image_topic'`. The script runs a while loop around the condition of roscore not being shutdown. I used CvBridge to convert OpenCV images to ROS messages. Then I initialised the camera using 'VideoCapture' following which the program published the messages to the fore mentioned rostopic.

2.2 Subscriber

The subscriber is a prime part of the process. At first I initialised the ROS node `'image_subscriber'`, following which I subscribed to the rostopic `'/image_topic'`. Then I created a CvBridge object to convert ROS messages to OpenCV images. Every time a message is received by this node, we call a callback function which converts the standard ROS message into OpenCV image for handling and processing the image. It is converted into a gray image and then to a canny image.

Then I displayed the processed image using OpenCV command 'imshow', following which I stacked the original image and the converted edge image horizontally together. I used a `camera_py.launch` file to run both the nodes. Then using `'rqt_graph'` command in a new terminal, I obtained the required ROS graph.

3 Conclusion

The publisher node reads the images using OpenCV and publishes them to a ROS topic, while the subscriber node subscribes to the ROS topic and processes the images further using OpenCV. This approach allows for powerful image processing capabilities in robotics applications.

Challenges

- identifying packages, nodes and rostopics
- opening the correct directory
- dealing with 'package not found' and 'directory not found' multiple times
- using 'source ./devel/setup.bash' every time before using rosrun in a new terminal
- finding/writing the required python code for the subscriber node

Canny Edge Images and RQT Graph

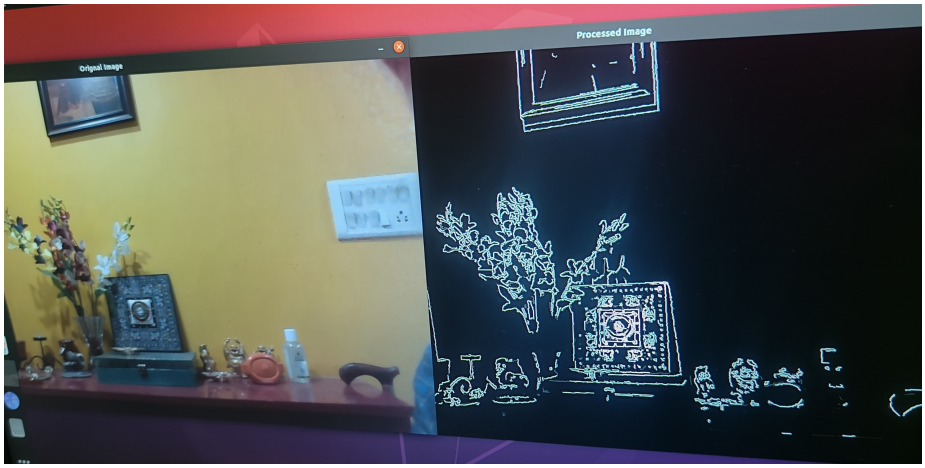

Figure 1: Edge Image 1

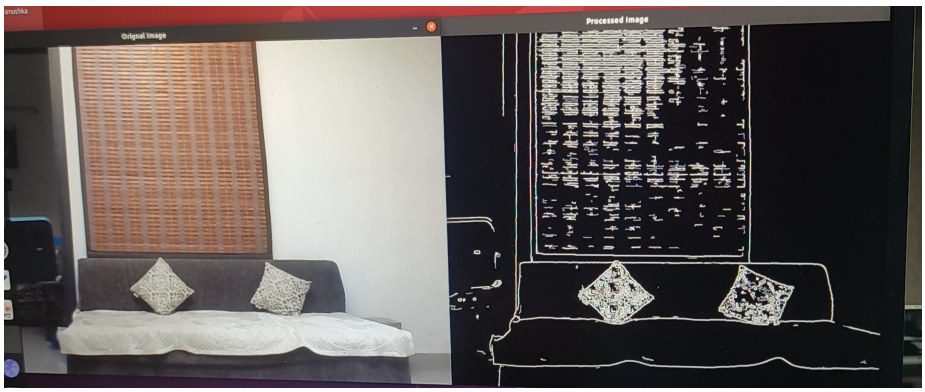

Figure 2: Edge Image 2

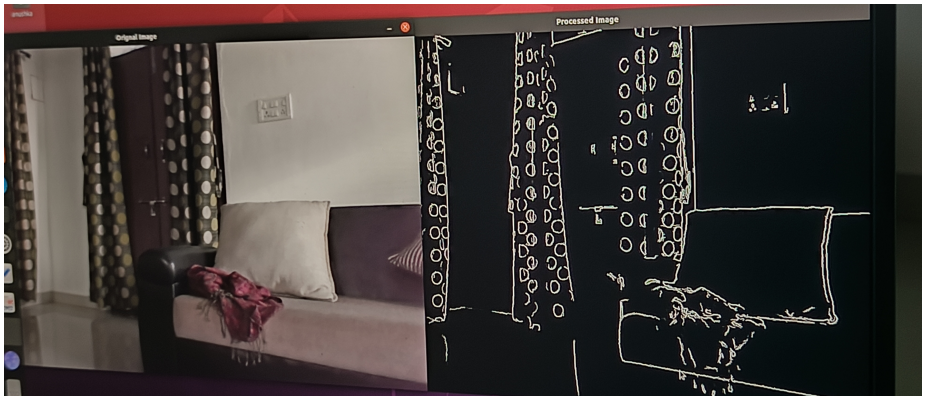

Figure 3: Edge Image 3

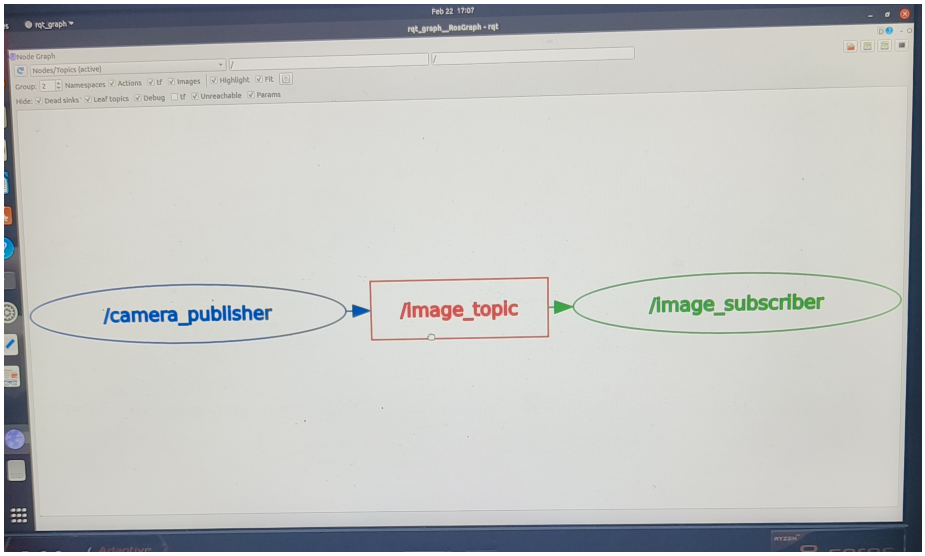

Figure 4: RQT Graph