Assignment 1 - Publishing/ Subscribing an Image and Canny Edge Detection

Atharva Tambat

April 30, 2022

1 Introduction

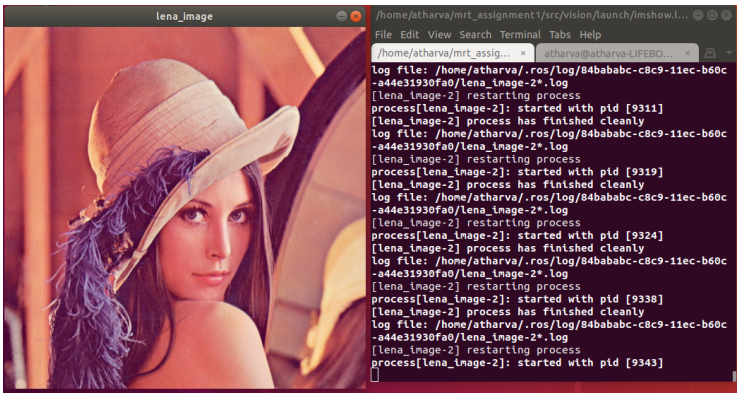
This is the first assignment in the MRT, which consisted of 3 tasks.

1. Task 1 - Make a Publisher that would publish images (instead of strings) to a rostopic using a cvbridge
2. Task 2 - Make a subscriber for the topic and for each image use canny edge detection method to find the edges in the image and display them side by side in a single window.
3. Task 3 - Make an rqt graph to visualize the connections between different nodes.

2 Task 1

The first step while completing task 1 was to understand how to write the python script for reading a still image file and displaying the file using OpenCV. For this, the package cv2 was imported into the python script. The following are the functions used -

1. cv2.imread('<filename>', <mode>) - reads the image into an object. The <mode> can be set to 0, 1 or -1 to get a grayscale, colour and unchanged image of the original image file.
2. cv2.imshow('<window heading>', <image object name>) - displays the image from the image object onto the window on the screen.
3. cv2.waitKey(<time in ms>) - commands the computer to show the image for the time specified in milliseconds before moving on to the next command.



Next step was writing a python script for reading a video file and displaying it on the screen. For this, again, cv2 package was imported. The following functions were used (except the ones already mentioned above):

1. cv2.VideoCapture(<Absolute path of the video file>) - to read the video into an object
2. <videoobject>.isOpened() - to check whether the video file opened successfully.
3. <videoobject>.read() - which returns values ret and frame: ret - boolean variable to check whether a frame is available and frame - a variable to store that frame.
4. <videoobject>.release() - once everything is done, release the video.
5. cv2.destroyAllWindows() - close all the frames at the end of the program.

Now.....ROS enters chat..... Next step was to add the extra code needed to create a publisher and publish the image to a topic, to the script that had already been written to display a video (with some slight changes).

For this some extra packages that were needed were:

1. rospy - to create a publisher and subscriber
2. cv_bridge - to convert the OpenCV Image to a ROS Image.
3. sensor_msgs.msg - to use some sensor related services and definitions (for eg: if one was using the video feed from the webcam)

The extra methods used in this script were:

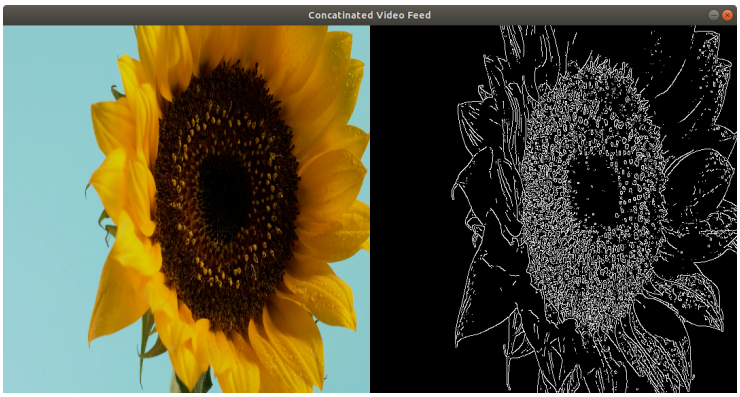
1. CvBridge() - for defining CV Bridge object.
2. rospy.init_node(...) - to declare this node.
3. rospy.Publisher(...) - to declare this node as a publisher.

4. rospy.Rate(...) - to define with what frequency should the program be run.
5. rospy.loginfo(...) - to publish node's status to the terminal for debugging.
6. <bridgeobject>.cv2_to_imgmsg(...) - to convert an OpenCV image to a ROS image message.
7. publisher.publish(...) - to publish the message (image) to the rostopic.
8. rospy.sleep() - to sleep just enough to maintain the desired rate.
9. rospy.spin() - to prevent ROS from exiting the node.

3 Task 2

This task was relatively shorter than the first - because all that was left was to subscribe to the topic. A rough outline of the steps that were followed are:

1. Importing necessary packages (previously mentioned packages - nothing new).
2. Define two functions - receive_message() and callback(data) - the former used for defining the node and declaring it as a subscriber (similar to what was done for the publisher) to the appropriate topic, and the latter used for the following:
   (a) Publish the status of the node for debugging.
   (b) Convert ROS Image to message to OpenCV Image.
   (c) Apply the canny edge method to find the edge of the images:
      i. Convert the image to grayscale using cv2.cvtColor(<name of image object>, cv2.COLOR_BGR2GRAY).
      ii. Apply the canny edge detection method using cv2.Canny(gray_image, 100, 300).
      iii. Convert the gray scale image to RGB to concatenate it with the original image, using cvtColor(...) function.
      iv. Concatenate the images (original and th one in which edge had been detected) using cv2.hconcat(...).
      v. Display the images on the screen using cv2.imshow(...).

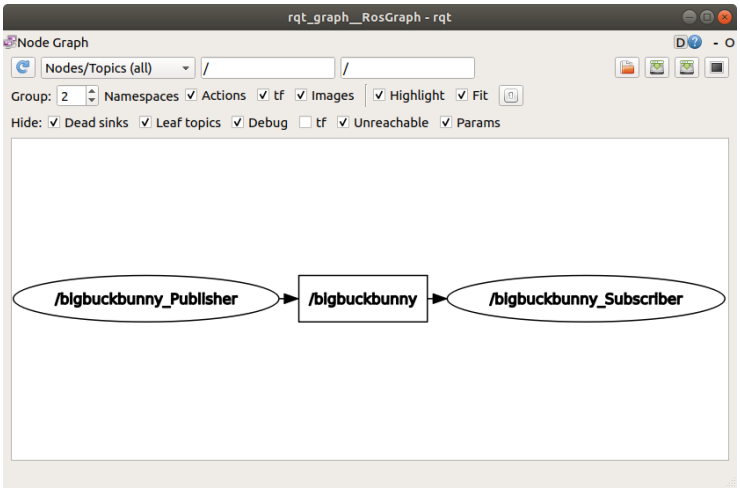


4 Task 3

The simplest one of the three - open new tab in the terminal and type the command:

rosrun rqt_graph rqt_graph

- Done!!!



NOTE: The subscriber and publisher nodes are name bigbuckbunny because I was using a 7 second clip of the big buck bunny - which proved to be too short to satisfactorily view edge detection. So, at the last moment I changed the video.