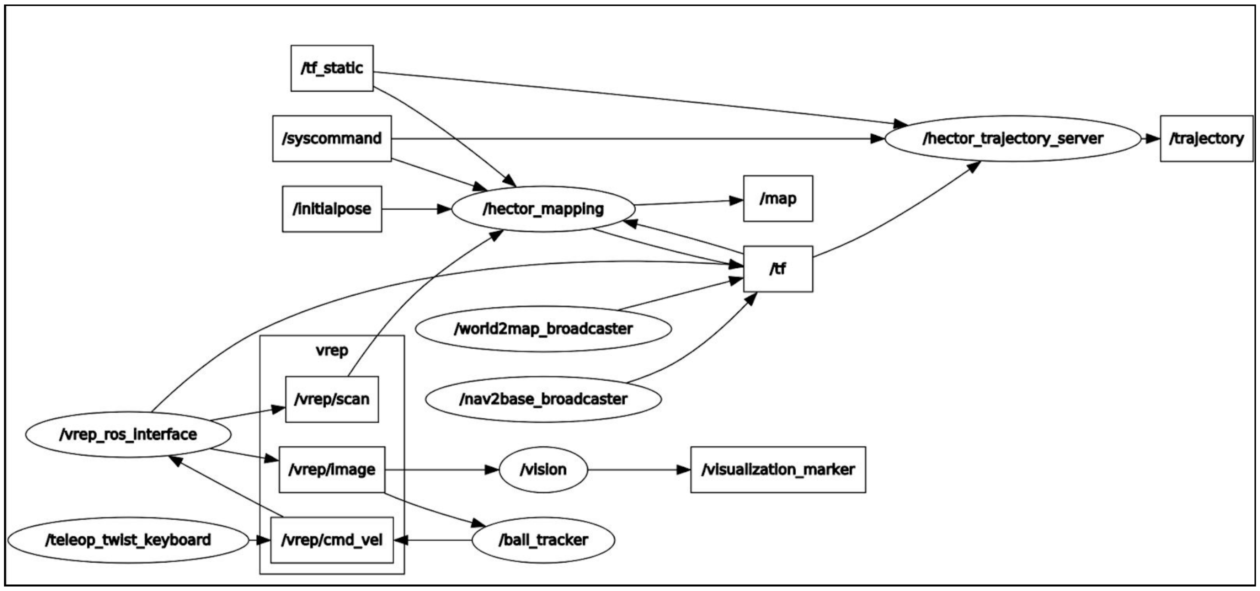
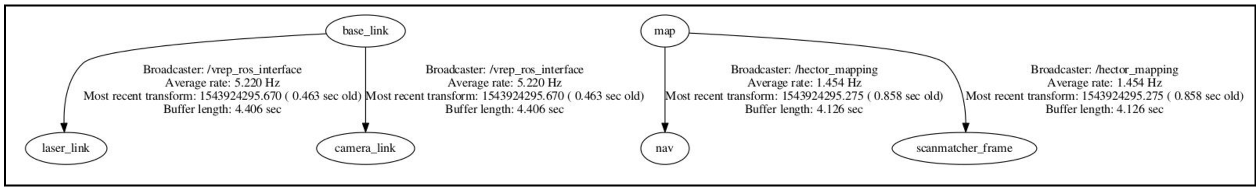
ELEC 4010K
Final Project Report

Thomas Narayana Swamy, WONG Yik Ben, YIU Yung Yu Andy

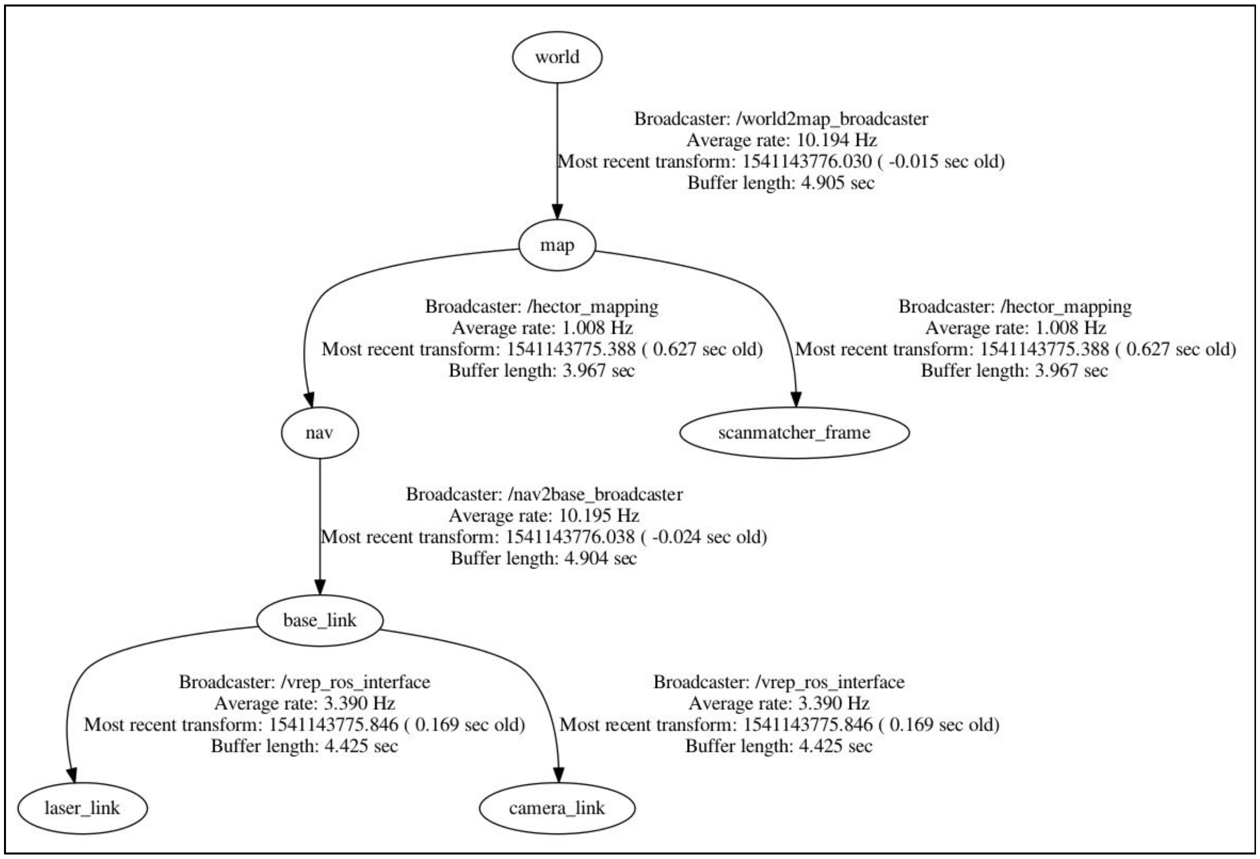
The figure below shows the ROS rqt_graph for our entire project. We can see how the entire system has been integrated and how each node communicates with one another using predefined rostopics.



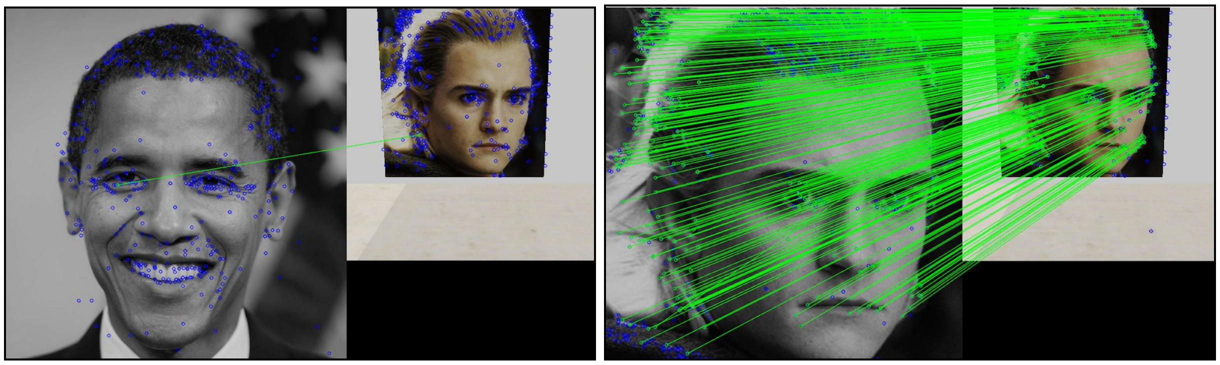
Below we see 2 tf trees. The first one shows us the tf tree for the robot frame and the hector mapping but as we can see the robot links are not connected to the map frame. Therefore, we modified the hector_mapping launch file to accommodate for this missing link by adding wold frame which can be see in the second tf tree below


base_link
Broadcaster: /vrep_ros_interface
Average rate: 5.220 Hz
Most recent transform: 1543924295.670 (0.463 sec old)
Buffer length: 4.406 sec
Broadcaster: /vrep_ros_interface
Average rate: 5.220 Hz
Most recent transform: 1543924295.670 (0.463 sec old)
Buffer length: 4.406 sec
laser_link
camera_link
map
Broadcaster: /hector_mapping
Average rate: 1.454 Hz
Most recent transform: 1543924295.275 (0.858 sec old)
Buffer length: 4.126 sec
Broadcaster: /hector_mapping
Average rate: 1.454 Hz
Most recent transform: 1543924295.275 (0.858 sec old)
Buffer length: 4.126 sec
nav
scanmatcher_frame

.

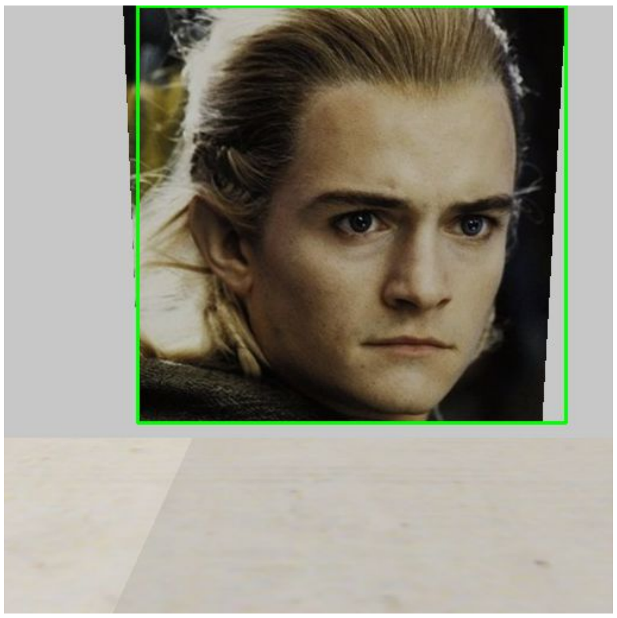

world
Broadcaster: /world2map_broadcaster
Average rate: 10.194 Hz
Most recent transform: 1541143776.030 (-0.015 sec old)
Buffer length: 4.905 sec
map
Broadcaster: /hector_mapping
Average rate: 1.008 Hz
Most recent transform: 1541143775.388 (0.627 sec old)
Buffer length: 3.967 sec
Broadcaster: /hector_mapping
Average rate: 1.008 Hz
Most recent transform: 1541143775.388 (0.627 sec old)
Buffer length: 3.967 sec
nav
scanmatcher_frame
Broadcaster: /nav2base_broadcaster
Average rate: 10.195 Hz
Most recent transform: 1541143776.038 (-0.024 sec old)
Buffer length: 4.904 sec
base_link
Broadcaster: /vrep_ros_interface
Average rate: 3.390 Hz
Most recent transform: 1541143775.846 (0.169 sec old)
Buffer length: 4.425 sec
Broadcaster: /vrep_ros_interface
Average rate: 3.390 Hz
Most recent transform: 1541143775.846 (0.169 sec old)
Buffer length: 4.425 sec
laser_link
camera_link

The following images show the result of feature extraction and comparison of using SIFT in OpenCV. All the images will store in the database in 2d array form and run the feature matching with the stored image in every frame, once the number of matched feature over threshold, the program will classify the image as the compared image.

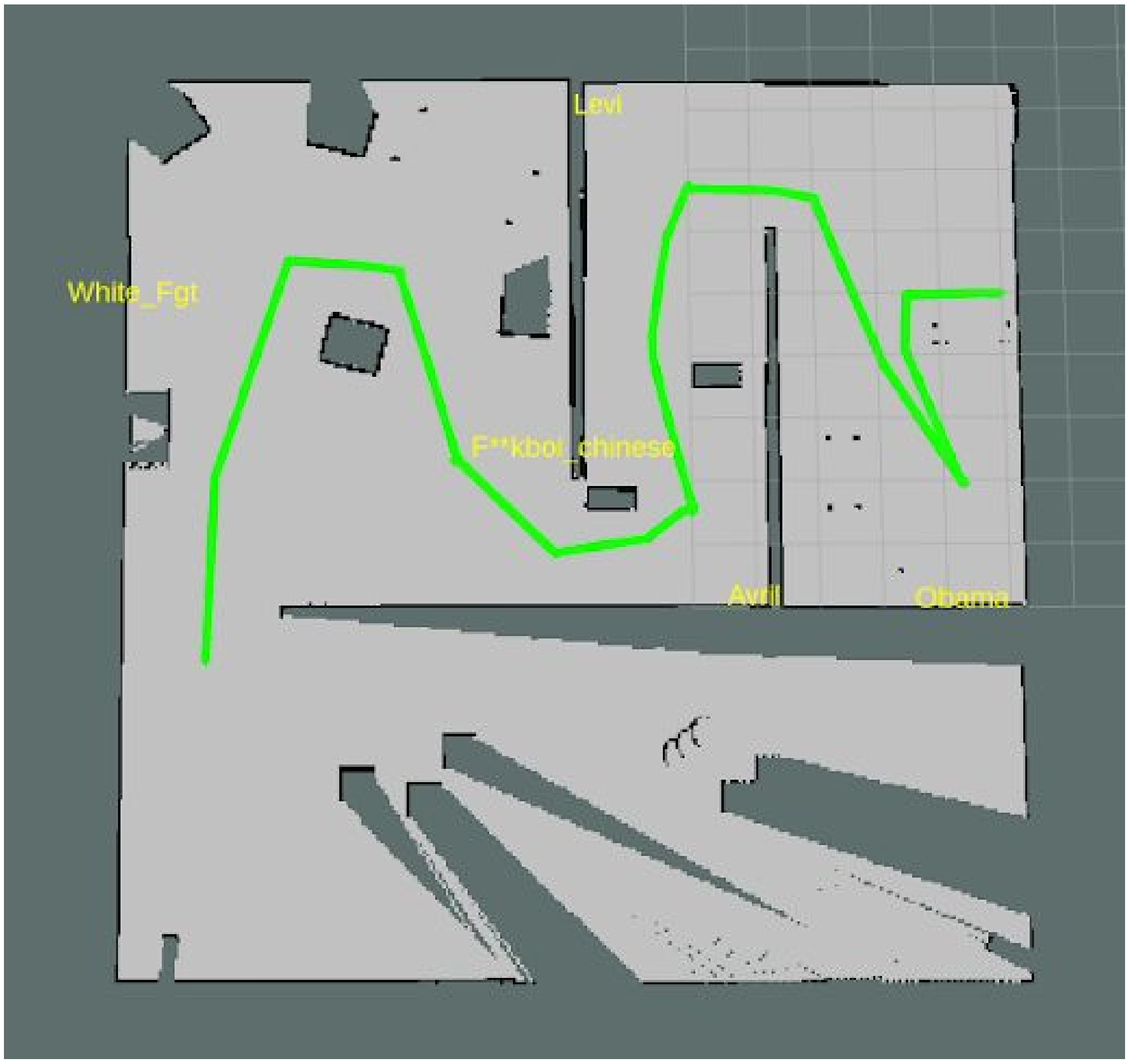


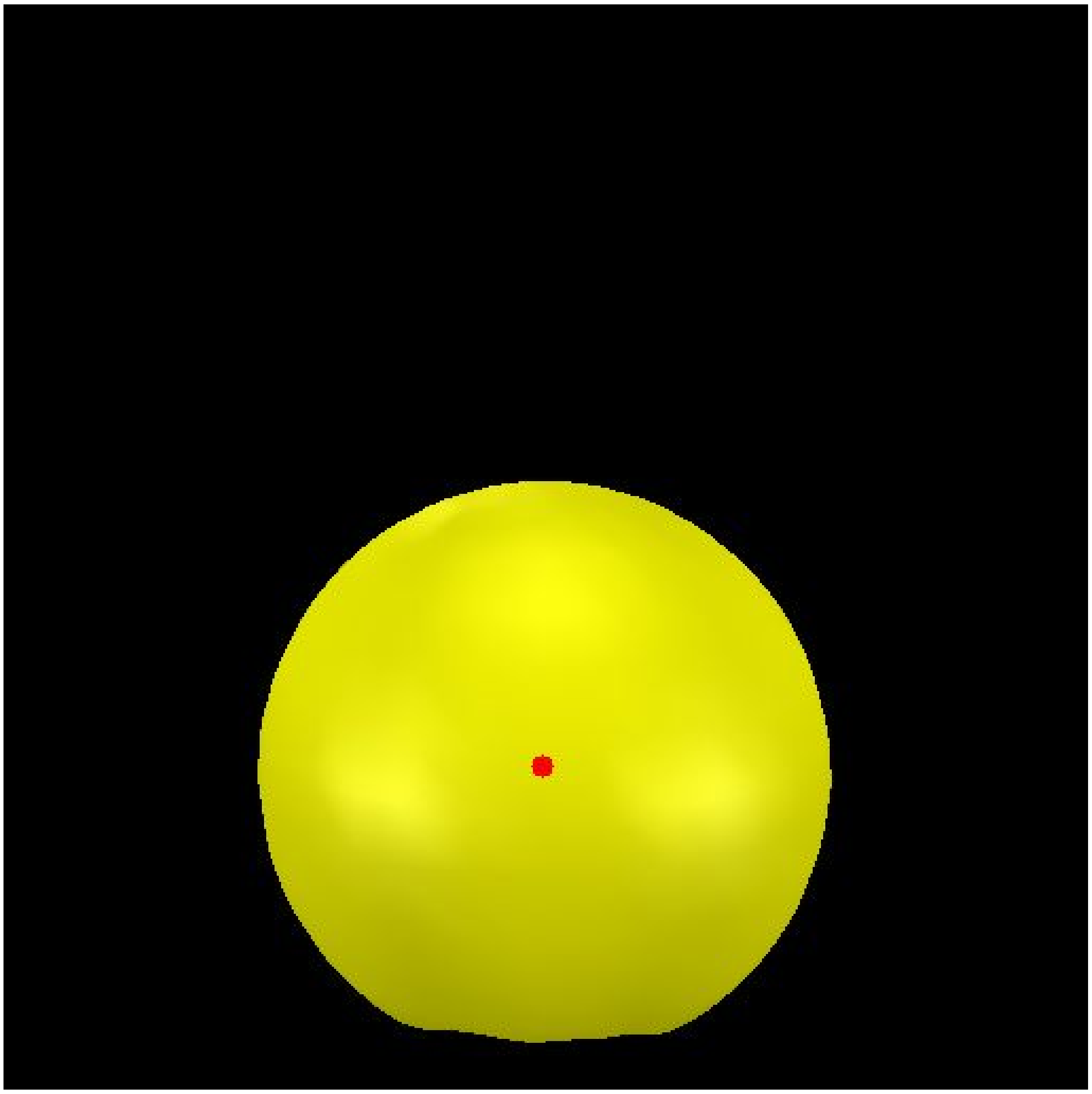
Then use the contour to mark the image with a green rectangle. Then compute the relative distance between image and robot with the camera's intrinsic parameters.

```python
def rviz_mark(contours, name):
    # find the biggest area
    c = max(contours, key=cv2.contourArea)

    x, y, w, h = cv2.boundingRect(c)
    # draw the book contour (in green)
    cv2.rectangle(cv2_img,(x,y),(x+w,y+h),(0,255,0),2)

    im_pix_h = h / 2
    im_theta = ((math.pi / 8) * h) / (512)
    dist_pix = (im_pix_h) / math.tan(im_theta)
    dist_x_real = ((dist_pix * 0.5) / im_pix_h)

    index = names.index(name) - 1

    mark_array[index].pose.position.x = dist_x_real
    mark_array[index].pose.position.y = ((x + w / 2) / 512)
    mark_array[index].pose.position.z = 0

    print(name + " " + str(dist_x_real))
    pub.publish(mark_array[index])

    # cv2.imshow('look', cv2_img)
    # cv2.waitKey(1)
```



The image below clearly represents the final results of the entire simulated code and environment. We can see that we were successfully able to map the simulation environment and identified the individuals for their respective images around the simulation environment. We used the SIFT feature matching algorithm mentioned above to identify and set the name of the marker, and used the camera and its inherent values to determine the image's distance relative to the camera_link and place the named marker at a point in the map as shown below. We further implemented a real time trajectory plotter to show the path taken by our robot in the simulation environment shown by the green line below.





The image above image is obtained from the camera in the ball tracker task after processing. The image is filtered by translating the color coordinate frame from rgb to hsv and removing the pixels which are not in the color bound. Then use a contour to locate the circle in the image and calculate the coordinate of centre and radius. Then the PID is done by inputting the error between the target radius and the target coordinate of centre. The radius error will reflect the vertical distance between the robot and the ball and the centre error will reflect horizontal distance. The robot is 2 DOF robot which one is for rotation and one is for translation. The PID for radius error is translation and PID for centre error is for rotation.