Indoor Mobile Robot Application:Sensing,Control and Planning

*Note: This report is for the final project of HKUST Course ROAS6000A

1st Yulin Li
dept. Robotics Institute
Hong Kong University of Science and Technology
Hong Kong, China
yline@connect.ust.hk

***Abstract*—This report presents the implementation details and results of the final project of ROAS6000A, upon which we will be able to manipulate an indoor mobile robot to navigate in unknown environments and gain information around via on-board sensors like camera and laser scan.With the sensor information,we will implement algorithms to do tasks like mapping,locolization,face recognition and visual object tracking,etc. By completing this project, we endows the mobile robot with moving, perception and cognition abilities.The whole experiment is conducted in simulation environment *VREP* and we interact with that via ROS interface.**

***Index Terms*—Mobile Robots,SLAM,Face Recognition,Visual Object Tracking**

I. Introduction

Autonomuous motion for robots is a prerequisite for many robotic applications. A mobile robot needs to perceive its environment in sufficient detail to allow completion of a task with reasonable accuracy. Particularly for mobile robots, the autonomous motion presents a very challenging problem because the robot has to know its position at all times relative to the environment. In order to know its position, the robot has to incrementally increase its knowledge of the environment which being relative knowledge, depends directly on the robot state.

With the current state belief of the robot itself and the environment,we could try many interesting tasks on the mobile system to gain the robot with more skills and thus make it smarter. The initial configuration is as shown in 1, the robot is set at a location in a room, and the robot don't know where it is and don't know what the environment looks like.

We divide the project into three parts, in the first part,we control the robot from keyboard command to move around and incrementally build the occupancy map of the environment based on laser scan inputs.After storing the environment map,we use ROS [1] package *amcl* from ROS Navigation Stack to do localization while moving arbitrarily in the room even the robot start from an unknown position and orientation. Then in the second part, we add a face detection and match

[1]The author is a 1st year PhD student in the Robotics and Autonomous System,HKUST

Fig. 1. Initial configuration of the robot in simulation environment.

module on the robot system, this module takes in on-board camera image input and marks where the face is relative to the room when it detects a face in the current image frame. Besides,we recognize who is the person on that image. In the last part, the robot tracks a moving ball in a certain area of the room. The yellow ball is moving in room D with a arbitrary path like the shape of number "8" .The robot detect the ball and automatically calculate the command twist to the robot to track the motion of the ball stably.

The rests of this report are arranged as follow:

- Section II illustrates the methods and experiment setup of each part in details to give overall a module descriptions.
- Section III will show the performance of each module and highlights some problems and difficulties encountered when implementing them.
- In the final section IV, I conclude my contributions upon this interesting project and give some directions and add-up points that could may worth be further explored.

II. Module Descriptions

A. Mapping and Locolization

Simultaneous localization and map building (SLAM) is a challenging problem in mobile robotics that has attracted the interest of more and more researchers in the last decade. Self-localization of mobile robots is obviously a fundamental issue in autonomous navigation: a mobile robot must be able to estimate its position and orientation (pose) within a map of the environment it is navigating in.

To obtain the map, we implement Rao-Blackwellized particle filter [2] to learn grid maps from laser range data, Rao-Blackwellized particle filters have been introduced as effective means to solve the simultaneous localization and mapping (SLAM) problem. This approach uses a particle filter in which each particle carries an individual map of the environment. The algorithm has a highly efficient and stable implementation version in ROS Package named **gmapping** for several ROS distributions. This package contains a ROS wrapper for OpenSlam's Gmapping. The gmapping package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

To use slam_gmapping, you need a mobile robot that provides odometry data and is equipped with a horizontally-mounted, fixed, laser range-finder. The slam_gmapping node will attempt to transform each incoming scan into the odom (odometry) tf frame.

To satisify the gmapping package requirement, I write a ROS node which maintains the odometry information of the mobile robot,namely the tf relationship of *odom* and *base_link*.Firstly I modify the LUA script in the VREP scene to publish the commanded wheel velocity for both right wheel and left wheel.

```
function velocity_callback(msg)
    -- This is the sub_velocity callback function
    -- sim.addStatusbarMessage('sub_velocity
    receiver: ' .. msg.linear.x)
    sim.auxiliaryConsolePrint(console_debug,'linear:
      ' .. msg.linear.x)
    sim.auxiliaryConsolePrint(console_debug,'
    angular: ' .. msg.angular.z .. '\n')
    vLeft = 6*msg.linear.x - 0.6*msg.angular.z
    vRight = 6*msg.linear.x + 0.6*msg.angular.z
    v = {}
    v['x'] = vLeft
    v['y'] = vRight
    v['z'] = 0
    simROS.publish(pub_wheel_vel,v)
    sim.setJointTargetVelocity(motorLeft,vLeft)
    sim.setJointTargetVelocity(motorRight,vRight)
end
```

The geometry information of the robot is as in 2 The initial robot base frame is align with the odometry frame and is set to:

$$q = \begin{bmatrix} \theta_z \\ x \\ y \end{bmatrix} = \mathbf{0} \tag{1}$$

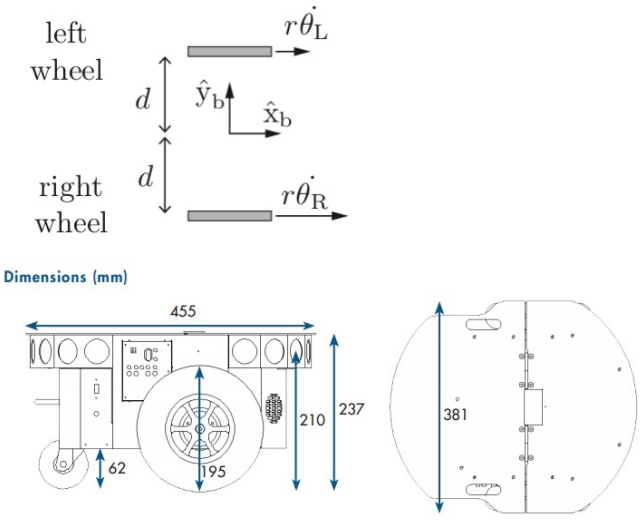

Fig. 2. Diff-driven car configuration

Then I use the left and right wheel velocity to calculate the incremental robot state *dq* in the interval *dt*, and update the current configuration of the robot with $q_{k+1} = q_k + \Delta q$. For the diff-drive robot ,we have:

$$V_b = F\Delta\theta = r \begin{bmatrix} -1/(2d) & 1/(2d) \\ 1/2 & 1/2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\theta_L \\ \Delta\theta_R \end{bmatrix} \tag{2}$$

Then the transformation in the car coordinate in this time interval could be calculated:

$$if\, w_z = 0, \Delta q_b = \begin{bmatrix} 0 \\ v_{bx} \\ v_{by} \end{bmatrix}$$

$$if\, w_z \neq 0, \Delta q_b = \begin{bmatrix} w_{bz} \\ (v_{bx}\sin w_{bz} + v_{by}(\cos w_{bz} - 1))/w_{bz} \\ (v_{by}\sin w_{bz} + v_{bx}(1 - \cos w_{bz}))/w_{bz} \end{bmatrix} \tag{3}$$

After getting the change in state in robot coordinate we need to transfer that into the world frame by premultiply the 2D spatial rotation matrix R_{sb}

$$R_{sb} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_k & -\sin\phi_k \\ 0 & \sin\phi_k & \cos\phi_k \end{bmatrix}$$

The tf tree visualized in rviz and rqt_tf_plot are respectively shown in 3 and 4after our odometry node running.

By sending the odometry information via *tf* system and subscribe to *Laserscan* topic, we could run gmapping node and adjust the parameters to get a relatively high quality map.

After building a map, I used *amcl* package to locolize the robot in the map. amcl is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses

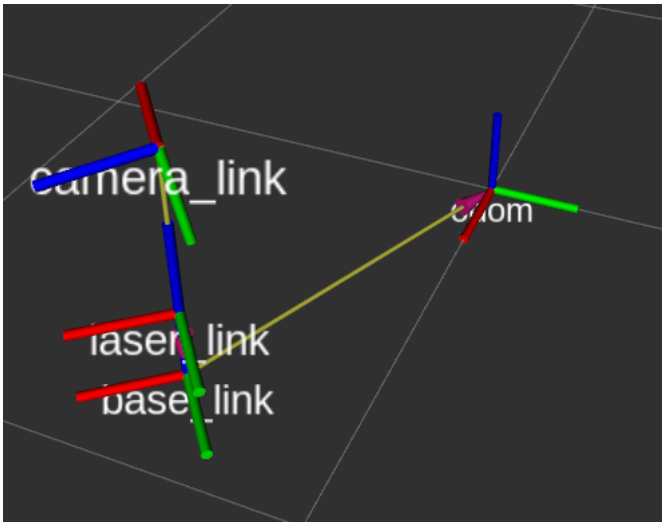

Fig. 3. Visualize tf in RVIZ

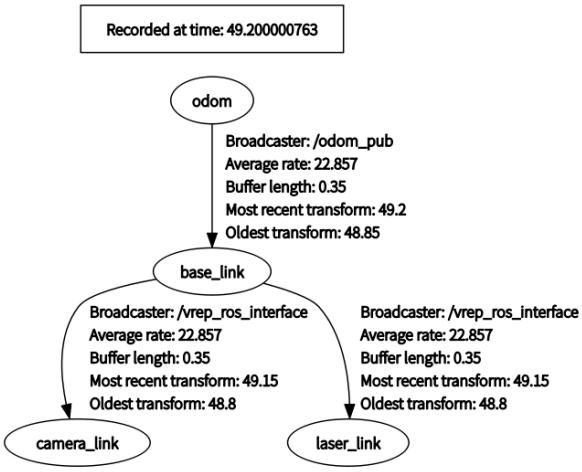

Fig. 4. tf plot

a particle filter to track the pose of a robot against a known map.Many of the algorithms and their parameters are well-described in the book Probabilistic Robotics [3], The user is advised to check there for more detail. In particular, we use the following algorithms from that book: sample_motion_model_odometry, beam_range_finder_model, likelihood_field_range_finder_model, Augmented_MCL, and KLD_Sampling_MCL.

B. Face Detection and Match

Face detection and recognition is a very interesting topic in computer vision, and current state-of-art algorithms are proved useful and stable in many real world applications.Facial recognition systems are employed throughout the world today by governments and private companies.

Generally, Facial recognition systems attempt to identify a human face, which is three-dimensional and changes in appearance with lighting and facial expression, based on its two-dimensional image. To accomplish this computational task, facial recognition systems perform four steps. First face detection is used to segment the face from the image background. In the second step the segmented face image is aligned to account for face pose, image size and photographic properties, such as illumination and grayscale. The purpose of the alignment process is to enable the accurate localization of facial features in the third step, the facial feature extraction. Features such as eyes, nose and mouth are pinpointed and measured in the image to represent the face. The so established feature vector of the face is then, in the fourth step, matched against a database of faces.

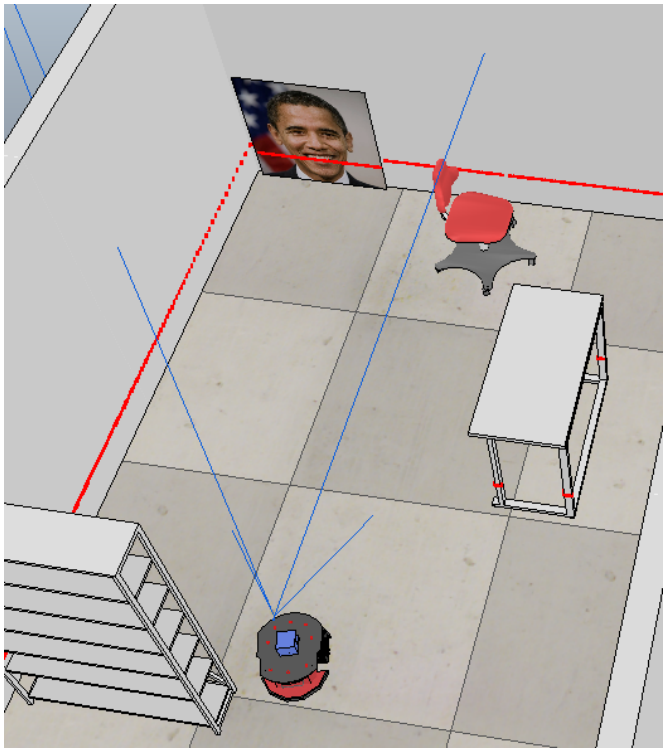

Fig. 5. Face Recognition in Simulation

This module is designed to read the image taken from simulation environment, and once a face is detected in the current frame shown in 5, the system will use a rectangle to highlight the face location and text the name of the recognition result in the image as well as locate the face in the world frame and add marker to that in rviz.

To accomplish this,I import the python library for face recognition:Library from github,and to set the training set image be different from the ones appear on the wall in the simulation environment to show the good generalization property of the algorithm.

Basically, the face location is detected by extracting the HOG pattern of the image and compare with the pre-trained nerual network using a lot of face HOG pattern images. Then

the algorithm uses a CNN to output face embedding,namely the face feature. last step is actually the easiest step in the whole process. All the algorithm do is find the person in our database of known people who has the closest measurements to our test image, a simple SVM classifier will do.

C. Visual Object Tracking

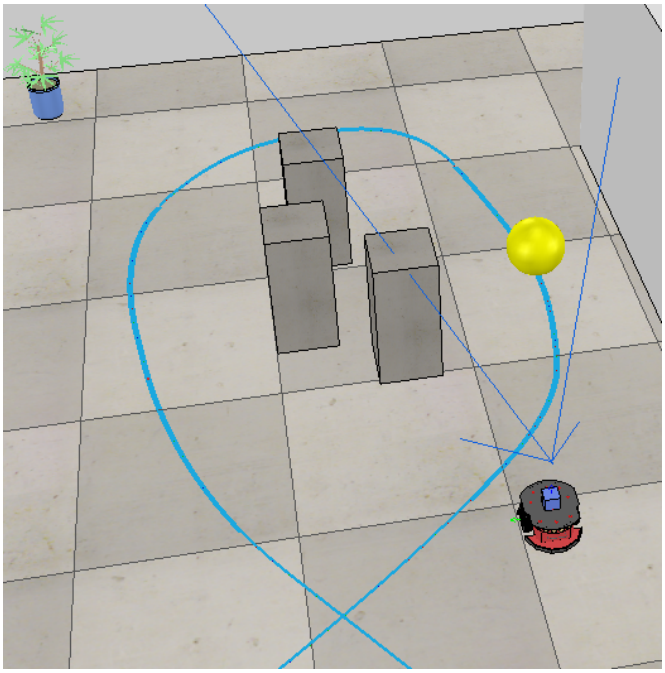

Fig. 6. Tracking Problem

Object tracking is the task of taking an initial set of object detections, creating a unique ID for each of the initial detections, and then tracking each of the objects as they move around frames in a video, maintaining the ID assignment.In other words, object tracking is the task of automatically identifying objects in a video and interpreting them as a set of trajectories with high accuracy.Object tracking is used for a variety of use cases involving different types of input footage. Whether or not the anticipated input will be an image or a video, or a real-time video vs. a prerecorded video, impacts the algorithms used for creating object tracking applications.

openCV object tracking is a popular method because OpenCV has so many algorithms built-in that are specifically optimized for the needs and objectives of object or motion tracking. Specific Open CV object trackers include the BOOSTING, MIL, KCF, CSRT, MedianFlow, TLD, MOSSE, and GOTURN trackers. Each of these trackers is best for different goals. For example, CSRT is best when the user requires a higher object tracking accuracy and can tolerate slower FPS throughput.

A shown in 6To automatically tracking the moving yellow ball for the robot, we need to first identify the yellow ball in each input image frame as in . There are a lot of ways to identify the yellow ball in the image. Here I used the HSV representation of the original image.HSV (for hue, saturation, value; also known as HSB, for hue, saturation, brightness) are alternative representations of the RGB color model, designed in the 1970s by computer graphics researchers to more closely align with the way human vision perceives color-making attributes. In these models, colors of each hue are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top. I do a Gaussian Blur to lower the noise after the color space transfer, then I set the lower and upper bound for the yellow in HSV image and get a bounding box of the yellow ball with the radius and center coordinate of the mask.

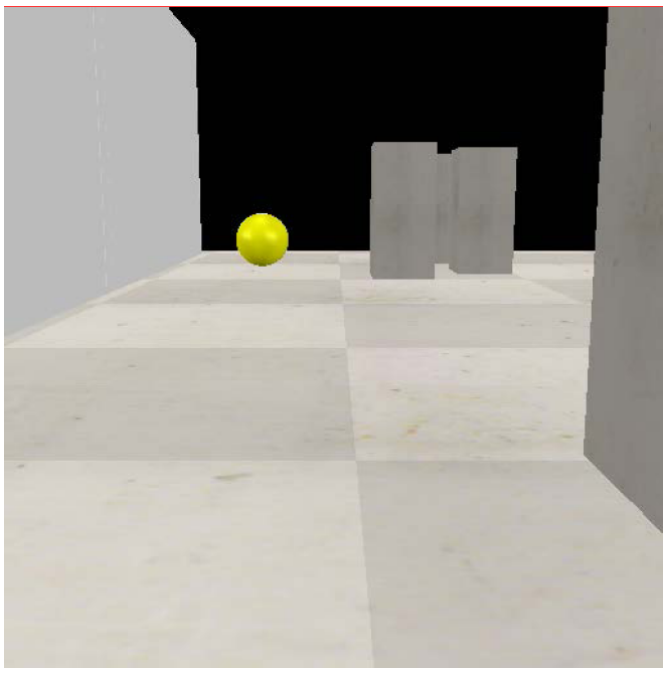

Fig. 7. Yellow ball in the image

As the image is taken from the camera frame,similarly we can say that the current express where the yellow ball is relatively to the robot(because the camera is statically attached to the robot and they share the same forward direction).so I set the vertical and horizontal offset at around the center of the image respectively and taking the bias of the current yellow ball location with the offset as error to formulate a simple PID controller. To be specifically,when the ball goes right or left, I give a calculated w_z corresponding to the rotation of the car and when the ball goes forward or backward, I change the linear velocity v_x to compensate that. Thus, the ball would be kept near the offset position in the image if the algorithm do a good job.

III. Results

In this section, I will show the result of my implementation of the three module respectively.

A. SLAM

After the odometry node be properly settled and the gmapping algorithm running. I can get a map incrementally

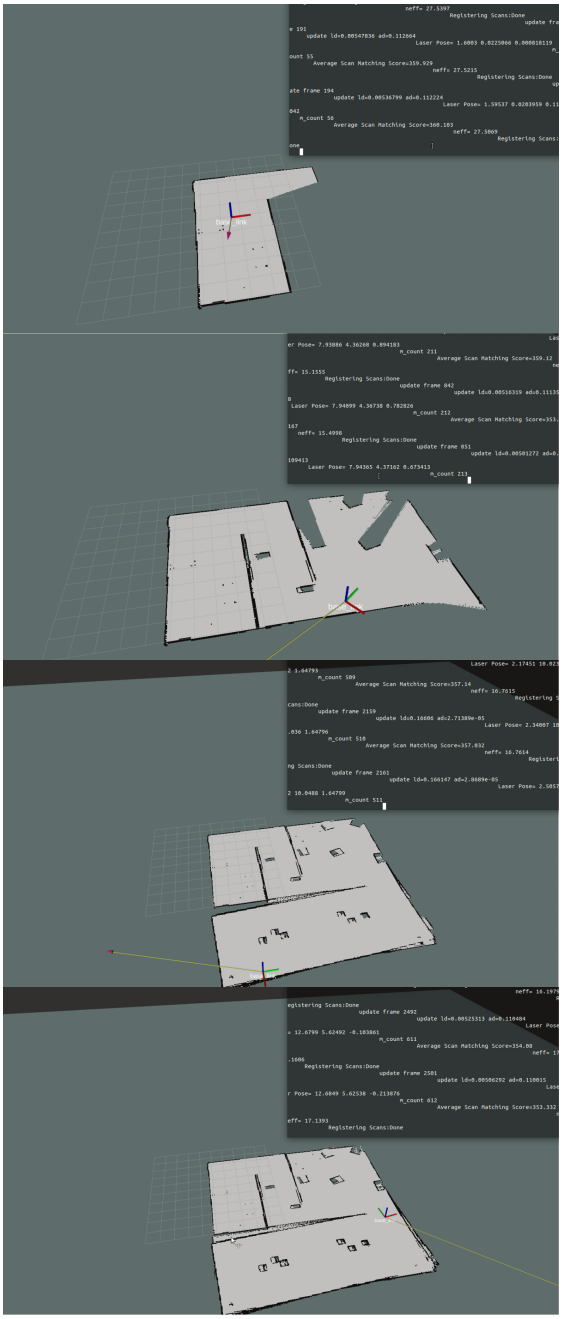

Fig. 8. Mapping Process

by traversing the mobile robot in the room like shown in 8.You can see that the information about the environment is added and modified as the mapping process goes on:the obstacles'shape be completed,the missing part of the room get closed. After this,we get a map using *map_server* package.Then I get the current estimation location of the robot in this known map using *amcl* and print out to screen the current room the robot believe it is in(like in my video).

B. Face Recognition

After converting the received ROS image message type to typical OpenCV image description. I implement the face detection and match algorithm as a ROS node, I write another node to send Marker in RVIZ to visualize the location of the image. I need to point out that the target images I used in the match process is different from thoes on the wall of the VREP scene. For the localization of the image, first I detect

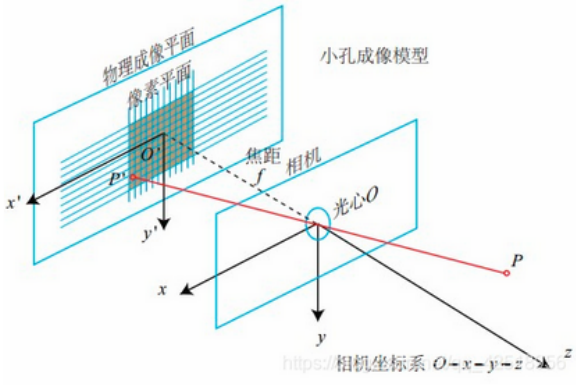

Fig. 9. Camera Model

the face and get the center of that in the image, and based on the camera model shown in 9, I calculate the angle of the center of face and then extract the data in laser scan range corresponding to that angle and get the absolute position. The recognition results are shown in 10:

C. Tracking Results

By using the method I proposed, I can get a relatively stable and efficient tracking result and you can see from the 11 that even though the plant comes into the sceen, the bounding box for the ball is still very accurate.

IV. Conclusions and Future Direction

This project has definitely solidate my foundation in programming robot with ROS and like shown in this report and my video,the results looks good! Basically,it is a very good practise for absorbing the knowledge in the course ROAS6000A and this project cost me almost two weeks from start on my own. From control of mobile robot to the perception and cognition ability of robot, although the tasks are presented in a relatively easy situation,it is still a very good overview for those aspects of robotics application.

Further, after getting the map and odometry tf,I have try using the ROS navigation stack to do some planning work,namely,given a desired location,the robot go there on its own using the pre-build map as the global costmap and use the laser scan data to construct for local costmap. When adding some dynamic object like walking human or other robots, the task becomes more challenging.

If a robot manipulator is articulated on top of the mobile basis, the so-called mobile manipulation.We can find more interesting topics to deal with,like how to grasp a object in a certain location based on visual input,etc.

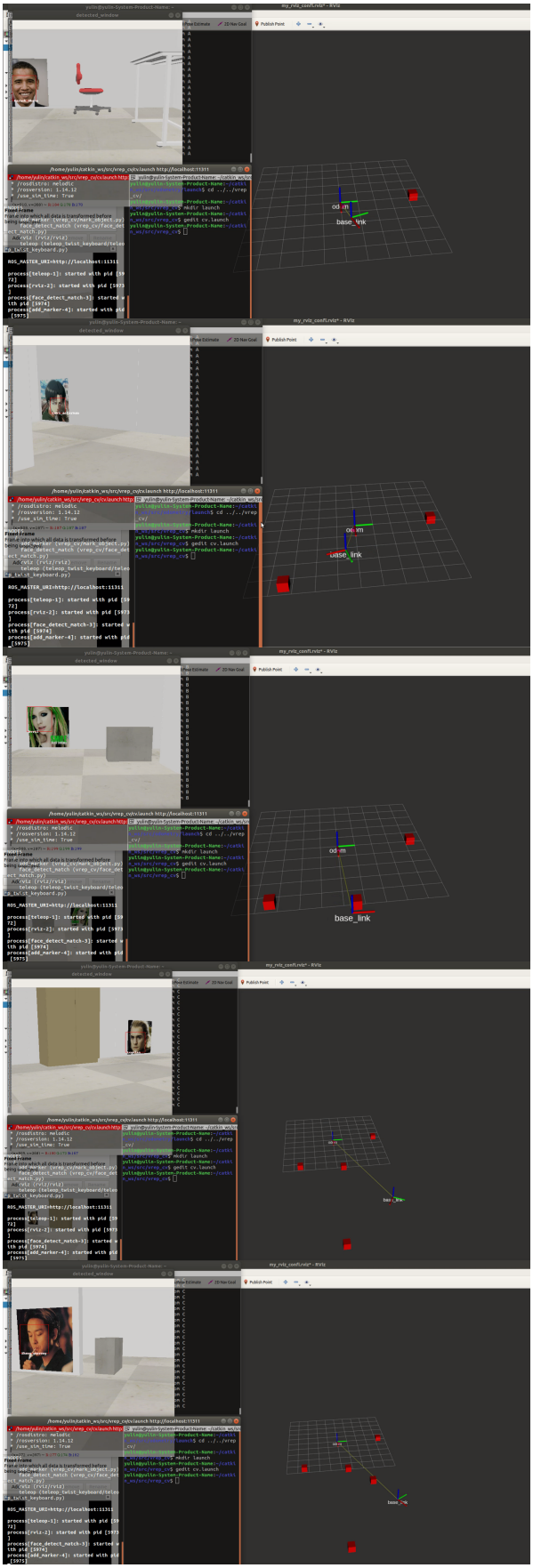

Fig. 10. Face Recognition Results

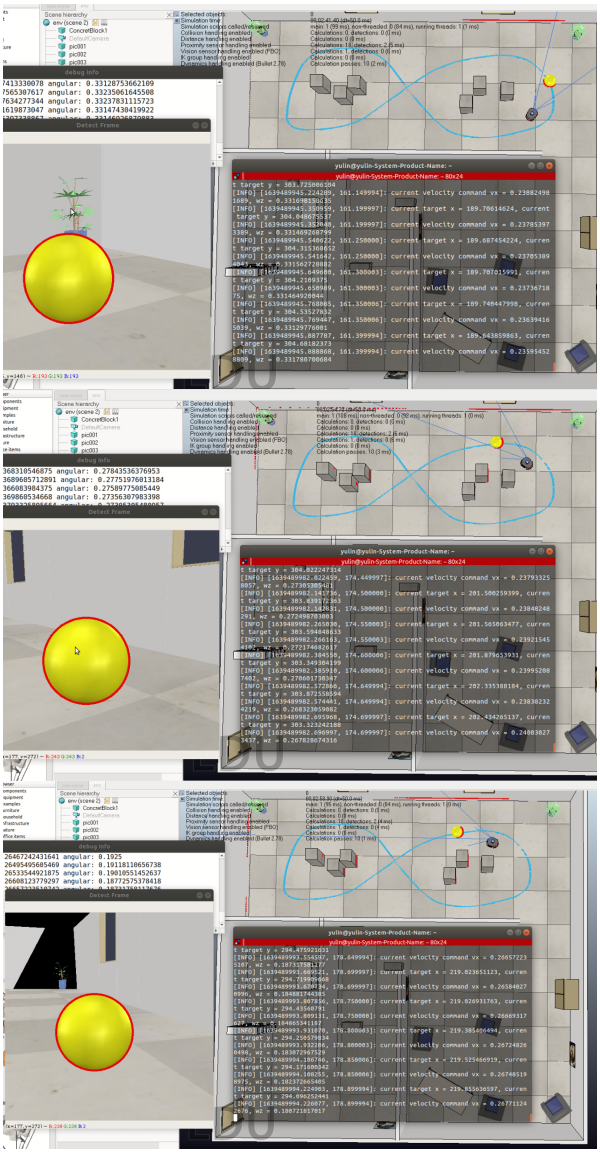

Fig. 11. Visual Object Tracking Results

REFERENCES

[1] ROS.org. (2021) Ros documentation. [Online]. Available: http://wiki.ros.org/

[2] C. S. Giorgio Grisetti and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[3] S. Thrun, *Probabilistic Robotics*. The MIT Press, 2005.