Lab 3: Perception

ME 597: Autonomous Systems

I. Introduction

One of the key challenges in mobile robots is the perception problem, but what is perception? Perception consists of making sense of the unstructured real world, in other words converting the real world into a more manageable version of it so that robots can work more efficiently with it. Remember that robots have a limited computation power, thus selecting what is important becomes a challenge and it is mostly driven by the application.

You have seen some of the most popular sensors used for robot perception in previous laboratories, such as ultrasonic sensors and LIDARs; however, you have not used one of the most important ones: the camera. In this laboratory, you will learn to use several libraries for image processing such as OpenCV.Although it does not have native support to work with ROS, cv_bridge will help us convert CV image objects into ROS messages that can be exchanged between nodes in a ROS application.

As part of this laboratory, you will utilize some algorithms for the robot to detect a moving red ball using the camera and to follow the red ball. In addition to OpenCV, you will have to work on the camera calibration, which will provide us with the required information to do other tasks such as marker detection and localization. In this lab you will learn how to use the ROS fiducial markers library, which provides with the tools to create, identify and locate tags with respect to the camera frame. This will allow the system to have a position reference system on top of which we can perform some control tasks.

II. Assignment

A. Camera Calibration and Fiducial Markers

- Over the next weeks we will be working on covering the fundamentals of monocular vision systems from a fairly classical approach. To start, we will be covering intrinsic camera calibration. Camera calibration, will take sometime, specially because we will be running a graphical interface remotely. Start by reading this tutorial.
- To prepare the RPi for all the things that you will need to run, it is necessary to install some libraries. Run the following commands to install the required dependencies for this lab.

```
$rosdep install camera_calibration
$sudo apt-get install ros-noetic-fiducials
$pip install opencv-contrib-python
$sudo apt install cairosvg python3-cairosvg
```

- Once all the dependencies are installed, it is necessary to do the calibration of you camera, for that, you need to first initialize the camera (running the provided `run_car.launch` file), and then run the calibration node.

```
$rosrun camera_calibration cameracalibrator.py --size 8x6 --square
   0.025 image:=/raspicam_node/image camera:=/raspicam_node --no-
   service-check
```

- In order to reduce the system load, it is recommendable to comment out from the provided launch file, the nodes that will not be used (motors, servos, line, leds and distance).
- Once the calibration node is running, you need to move in all directions the checkerboard image that was given to you. always ensuring that the checkerboard is detected by the calibration algorithm. Once all the status bars of the calibration are green, it is the signal that we have enough samples to perform the calibration. At that moment, the calibration button will change its color, which will allow to proceed to the camera calibration. After that, the other two buttons will also change its color, you need to first hit save and then commit. This will save the calibration into a file that the `raspicam_node` will use.

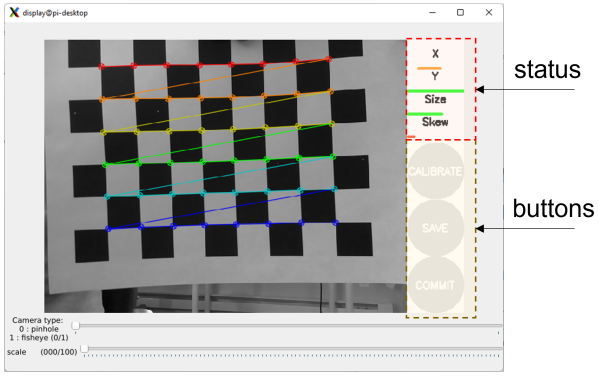

Fig. 1: Camera calibration interface

- Copy the generated calibration file `~/.ros/camera_info` into the `~/aut_sys_ws/src/aut_sys/src/raspicam_node/camera_info` folder.
- Now that the camera is calibrated, we will use fiducial markers. These markers are similar to QR codes, and they can be detected by the camera, providing a mechanism to determine translation and rotation of the marker with respect to the camera. Each of these markers are tied to an ID, which can be used to filter out the markers that we are interested in. The node in charge of this detection is `fiducials`.
- This node publishes the detected fiducial markers into the topic `/fiducial_transforms`. You can find more details of the message structure in the following link. This has to be enough to know which library has to be imported to implement the required subscriber that catches the fiducial marker that you need.
- The goal of the this section of the lab is to use the fiducials node to drive the UCTRONICS robot towards a fiducial marker and park the robot (1m,1.5m) away from the fiducial marker as shown in Fig.2.

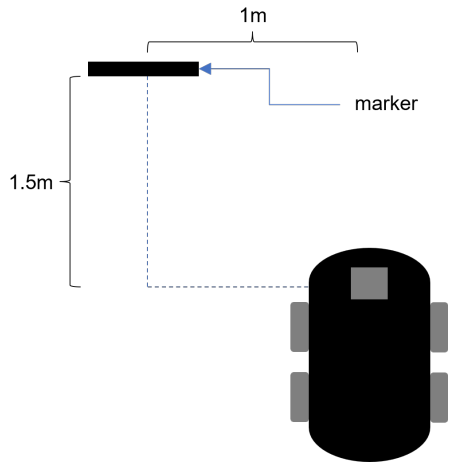

Fig. 2: UCTRONICS car and fiducial marker

B. Ball detection and tracking (Gazebo)

- Create a ROS package named `lab_3_pkg` with dependencies `rospy` and `std_msgs` in `~/catkin_ws/src`
- Download object_tracking.launch and place it in the `launch` folder.
- Create `/scripts` directory and download ball_controller.py and place it in the `scripts` folder.

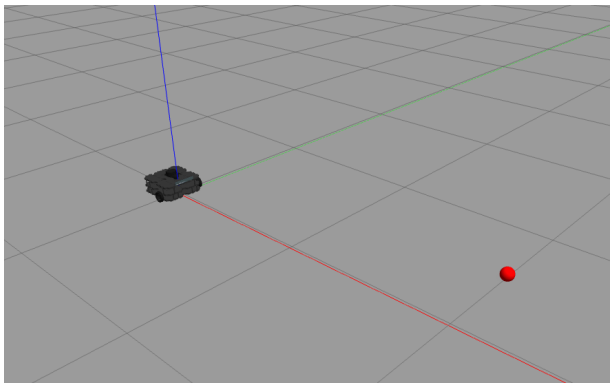

Fig. 3: Lab 3 environment

- Run the following commands to install `pynput` library

```
sudo apt install python3-pip
pip install pynput
```

- Compile the package and `run object_tracking.launch`. You can use WASD keys on the keyboard to control the position of the red ball.
- Create a file named `track_red_ball.py` in `/scripts` directory and use what you learned from the vision use case lecture to track and approach to the red ball.
- Subscribe to `/camera/rgb/image_raw` to get the image from the camera.
- Run this script using `rosrun lab_3_pkg track_red_ball.py`.

III. Deliverables

1) Verify you have completed all the assignments.

2) The python file created to implement the object tracker.
3) The python file created to implement the fiducial based controller.

IV. RUBRIC

- 50 pts - Ball Detection and Tracking.
  - 20 pts - Successful detection of the red ball.
  - 20 pts - The vehicle follows the ball without loosing it.
  - 10 pts - Code functional and properly commented.
- 50 pts - Camera Calibration and Fiducial Markers
  - 20 pts - Successful detection of the fiducial markers assigned to you.
  - 20 pts - The UCTRONICS car drives towards the fiducial marker and parks at the right position.
  - 10 pts - Code functional and properly commented.