EE346 - Mobile Robot Navigation and Control

Winter 2022
Laboratory #3 (5%)
Due Date: Wednesday April 6, 2022

Camera Calibration and AR Tag and ArUco Marker Recognition

Objectives

- Learn how to calibrate a camera
- Study pose estimation with AR tags
- Study pose estimation with ArUco markers

Procedure

In this lab, you will first learn to how to calibrate a camera on your laptop (either built-in or a USB webcam) using the ROS package camera_calibration. Then you will proceed to study how to detect AR tags and estimate camera pose with respect to an AR tag using the ROS wrapper for Alvar, ar_track_alvar. Finally, you will study how to detect ArUco markers and estimate camera pose with respect to these markers using the ROS package aruco-ros.

Part I: Camera calibration (30%)

In this part of the lab, you will calibrate the monocular camera. Use the check board pattern that is provided to you, and follow the instructions at http://wiki.ros.org/camera_calibration for running the camera calibration node. Make sure that your camera node is properly publishing images to the topic /camera/image_raw. Study the tutorial at http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration, to perform the actual camera calibration. Complete the steps in the tutorial, and save the calibration results in a yaml file (typically named camera.yaml) for use in the next two parts. Inspect camera.yaml to understand its contents, which define your camera's characteristics: image resolution, calibration matrix, distortion coefficients, rectification matrix, and projection matrix.
ROS camera calibration package is based on OpenCV, and this page explains further details on if you are interested: https://docs.opencv.org/master/d4/d94/tutorial_camera_calibration.html.
Also you can read this tutorial to understand the camera calibration algorithm and process: https://learnopencv.com/camera-calibration-using-opencv/.

Upon completion, show the calibration parameters in the saved yaml file to a TA, and explain what the numbers mean.

Part II: AR tag detection and pose estimation (30%)

In this part of the lab, you will study how to recognize and perform pose estimation of AR tags, a fiducial (reference) marker system to support augmented reality (AR). We will use the popular ROS package, ar_track_alvar, a "wrapper" around the AR tag library called Alvar, in our experiments. First study the Wiki page: http://wiki.ros.org/ar_track_alvar, to install the package.

You can either print your own AR tags for numbers from 0 to 9 (10 different numbers). With the camera node running, run the node ar_track_alvar and display/visualize the two topics published by the ar_track_alvar node: the first on the topic of visualization_marker indicates which AR tag (of the 18) is being recognized, and the second on the topic of ar_pose_marker shows the pose of the camera with respect to the AR tag in terms of a tf message. Make sure that you are able to recognize the AR tags correctly and verify the pose in the tf message. Note that the Wiki page above is for a different robot called PR. To modify the launch file for the ar_track_alvar node to work for this lab, refer to the blog at http://ros-robotics.blogspot.com/2015/04/recognize-ar-tags-with-ros.html for additional information.

Upon completion, show the operation of the AR code detection by capturing images by the camera on your computer, placing one of the AR tags in front of the camera, and printing on the screen both the AR tag number and the relative pose from the camera to the AR tag. Make sure you can explain the Pose message that describes the pose.

Part III: ArUco marker detection and pose estimation (40%)

In this part of the lab, we still study how to perform camera pose estimation with ArUco markers. First, study the principles behind ArUco markers by reading the following two pages at:
https://automaticaddison.com/how-to-create-an-aruco-marker-using-opencv-python/, and
https://automaticaddison.com/how-to-detect-aruco-markers-using-opencv-and-python/

Then create and detect ArUco markers with the two programs on the above two pages:

generate_aruco_marker.py and detect_aruco_marker.py.

Then Install the ROS package aruco_ros by following instructions on the page at:
https://github.com/behnamasadi/aruco-ros

Duplicate the result in the video at the end of the page for ArUco maker ID 701. Note that, for the experiment to work, you need to

(a) Create and print an ArUco marker with ID 701to complete this experiment,
(b) Create the two launch files in your own catkin workspace, and
(c) Modify pixel_format parameter in the launch file usb_cam_stream_publisher.launch from "mjpeg" to "yuyv", and
(d) Set the markerSize argument in aruco_marker_finder.launch according to the real size of the marker you print in (a).

Echo marker pose on the topic /aruco_single/pose, and make sure that it is accurate.
Demonstrate your experiment to a TA and be prepared to answer questions.

Marking

If you are able to complete all three parts before the end of the lab on April 6, you will receive 100%. If you are not able to complete any parts of the lab by the due date and time, you will get a 20% penalty of those parts, and an additional 20% for each day of delayed demo.