Ground truth localization

183DA: Design of Robotic Systems I - Lab 3 and 4

This document describes how Aruco tags are used to localize a robotic agent in an environment. In particular labs 3 and 4 require the estimation of a robot's state. The tool described below records the ground truth position and orientation of the robot in real time. To highlight the functionality of the system the Aruco tags will first be introduced followed by instructions for operating the localization system itself.

Aruco tags

In general, it is hard for a computer vision system to identify objects in the world due to irregularities in shape, dynamic lighting and many other factors. To make this task easier researchers have developed identification tags which are easily identifiable. These tags come in the form of an n x n binary grid of black and white blocks, the pattern of which unique describes the tag. In this sense, they are much like QR codes except smaller:

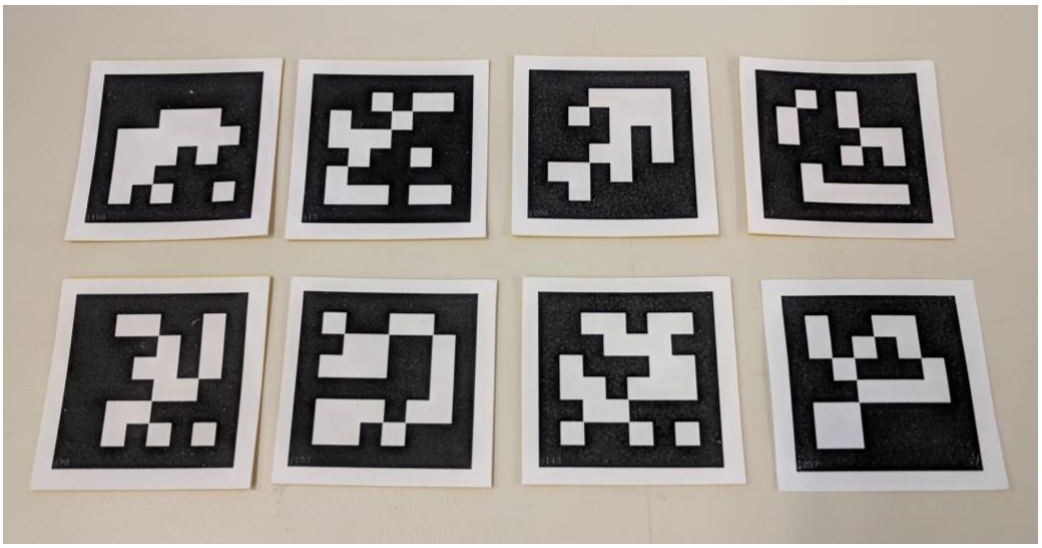

Figure 1: Aruco tag examples

The asymmetric nature of the pattern is a beneficial property since the tag position and orientation can be detected with ease. Furthermore, knowing the physical size of the tag then enables the calculation of the transformation matrix to the tag and it can therefore be localized with respect to the camera.

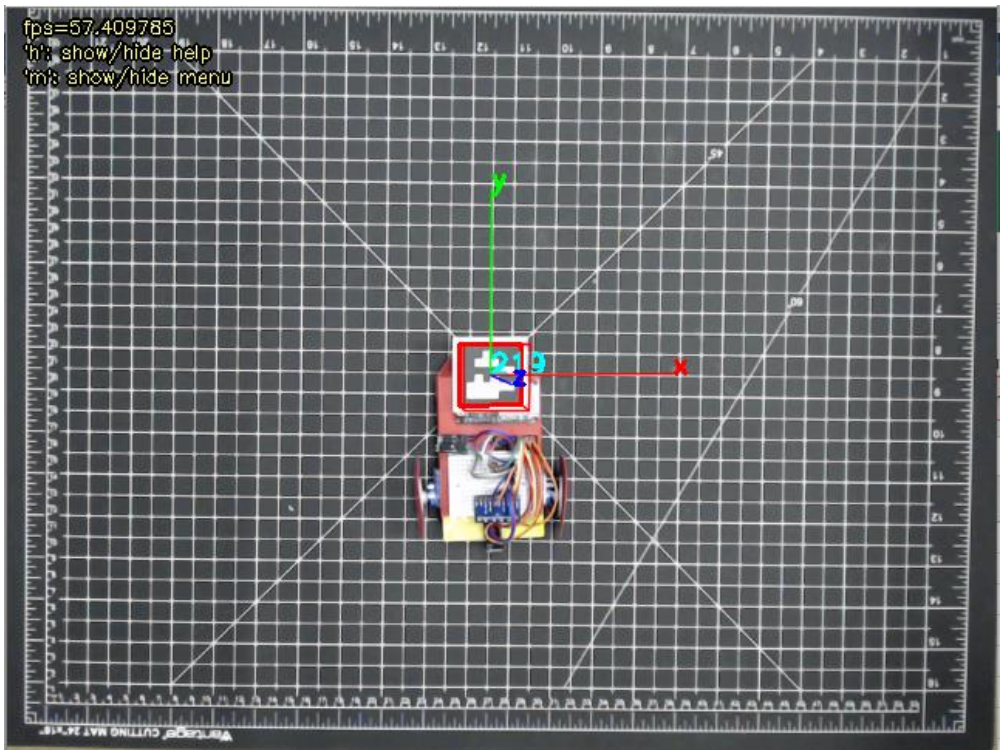

Figure 2: Localization example using an Aruco tag

Multi agent localization is also supported through the unique tag patterns:

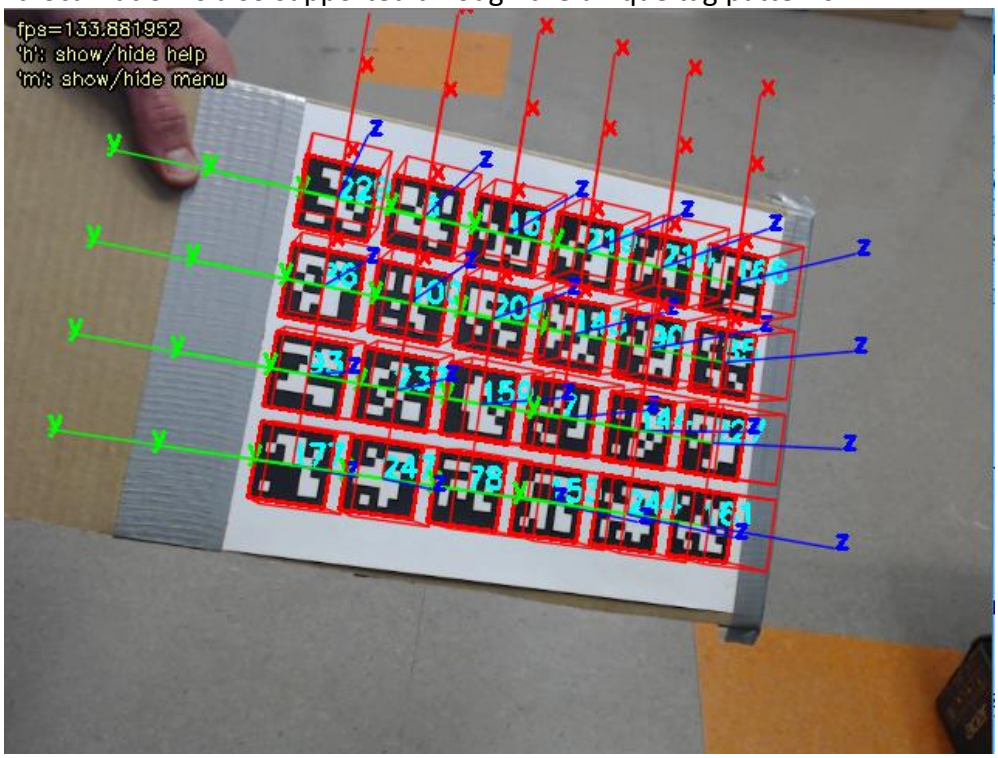

Figure 3: Multi agent localization example using a grid of tags

Localization system

The environment is setup as shown below. The positioning of the Aruco tag in the environment enables the camera to be positioned at x = 0, y = 0 with respect to the tag (in other words directly above it) such that the readings produced by the system represent the ground truth measurements. Once the camera has been positioned correctly the tag can be obstructed such that it does not interfere with other the robot tag. (Note: I will center the camera beforehand, but it should be checked as things are likely to move)

The operation procedure is as follows:

- Prepare the vehicle:
  - Attach an Aruco tag to the robot taking note of the offset of the tag with respect to the robot frame.
- Run the aruco_test.exe with the following command:
  - **aruco_test.exe live:0 -c C:\Users\LEMUR\Downloads\aruco_304_x64_vc14\calibration\camera_cal1.yml -s 0.036 -d ARUCO_MIP_36h12**
    - The first argument references the camera as an input device
    - –c refers points to the calibration yml file for the camera
    - –s is the physical size of the aruco tags
    - –d is the type of tag that will be detected
  - This will launch two windows: A live scene with superimposed tag frames and a black and white discrimination scene.
  - The command line gives pose estimation for the tags in the scene. The information is displayed in a format shown in the table below.
- Ensure that the camera is zeroed:
  - Move the ENVIROMENT instead of the camera (this is easier and more accurate) such that the tag in the environment has readings x = 0, and y = 0
- Position robot vehicle directly above the environment tag such that it starts at the origin with respect to the camera
- Begin experimentation:
  - The readings on the screen give a live update of the position of the robot
- Once complete stop the program
- Access the csv log of the session:
  - The csv is saved at **C:\Users\LEMUR\Documents\localization_data\example.csv**
  - Note the csv is overwritten with each run

Table 1: Terminal and CSV file data output structure

Timestamp	Tag ID	Txzy	Tx	Ty	Tz	Rxyz	Rx	Ry	Rz

Timestamp: timestamp at which the reading was recorded, the format is is microseconds since the epoch

Tag ID: Unique tag ID which can be found in the bottom left hand corner of the tags
Txzy: Transform label describing the next three columns, **units in meters**
Tx: Transform with respect to x
Ty: Transform with respect to y
Tz: Transform with respect to z
Rxzy: Rotation label describing the next three columns, **units in radians**
Rx: Rotation with respect to x
Ry: Rotation with respect to y
Rz: Rotation with respect to z

NOTE: The Ty reading has a reversed sign to what is expected. This should be corrected before comparison.

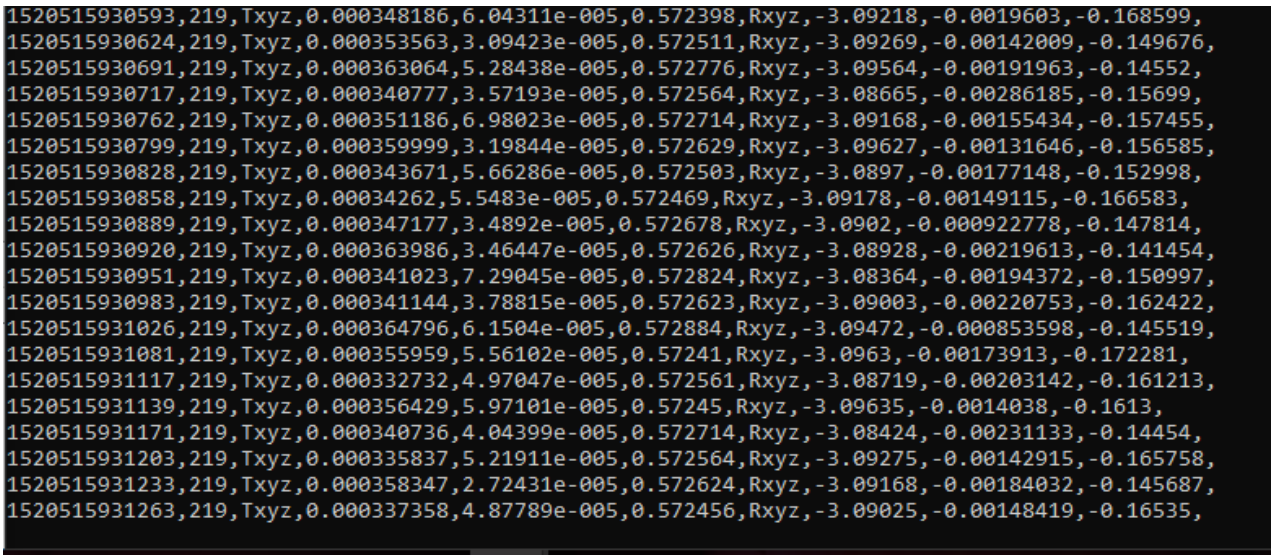

Figure 4: Typical command line output