MEAM 620 Project 2 Phase 1

Due: Thursday, March 28th, 2019

For this phase you will implement a vision based 3-D pose estimator which estimates position and orientation of the quadrotor based on AprilTags [1].

1 Vision Based Pose Estimation

1.1 Environment

The data for this phase was collected using a Nano+ quadrotor that was either held by hand or flown through a prescribed trajectory over a mat of AprilTags, each of which has a unique ID that can be found in `parameters.txt`. Figure 1 shows the layout of the AprilTag mat. The tags are arranged in a 12 x 9 grid. The top left corner of the top left tag should be used as coordinate (0, 0) with the X coordinate going down the mat and the Y coordinate going to the right. Each tag is a 0.152 m square with 0.152 m between tags with the exception of the space between columns 3 and 4, and 6 and 7, which is 0.178 m. Using this information you can compute the location of every corner of every tag in the world frame.

1.2 Calibration

The intrinsic camera calibration matrix and the transformation between the camera and the robot center are given in `parameters.txt`. Two photos (`top_view.jpg` and `side_view.jpg`) are included to visualize the camera-robot transform. You will need to transform your camera-based pose estimate from the camera frame to the robot frame so that you can compare it against the ground truth VICON data.

1.3 Pose Estimation

The data for each trial is provided in a `mat` file. The file contains a struct array of image data called `data`, which holds all of the data necessary to do pose estimation. The format of image data struct is as follows:

1. Time stamp (`t`) in seconds.
2. ID of every AprilTag that is observed in the image (`id`).
3. The center (`p0`) and four corners of every AprilTag in the image. The corners are given in the order bottom left (`p1`), bottom right (`p2`), top right (`p3`), and top left (`p4`), see Figure 2. The i^{th} column in each of these fields corresponds to the i^{th} tag in the ID field and the values are expressed in image coordinates.
4. Rectified image (`img`).
5. IMU data with Euler angles (`rpy`), angular velocity (`omg`), and linear acceleration (`acc`).

Note that for some packets no tags are observed. In these cases you will not compute the pose for the packet but will return empty arrays (see code header for more information). The rectified images and IMU data are not necessary for this phase, but we keep them for the sake of data format consistency between this phase and future phases. You must compute the measured pose of the Nano+ for each packet of data using the camera calibration data, the corners of the tags in the frame, and the world frame location of each tag.

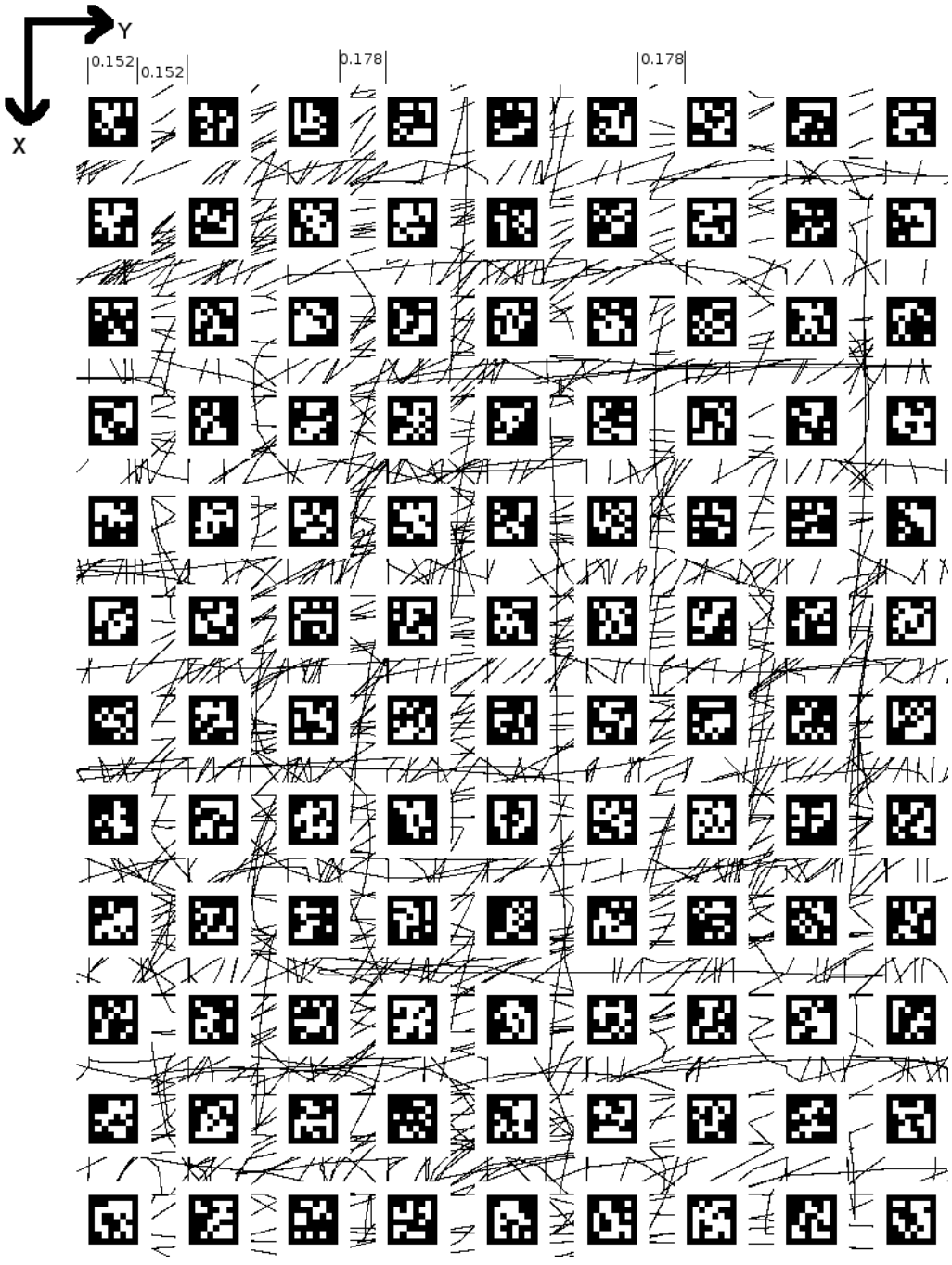

Figure 1: The AprilTag mat

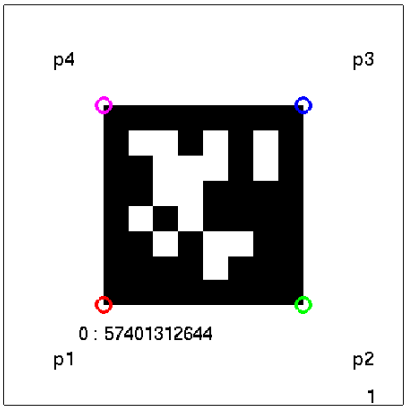

Figure 2: Corners of the AprilTag

The `mat` file also contains VICON data taken at 100 Hz, which will serve as our ground truth measurements. The VICON data is stored in two matrix variables, `time` and `vicon`. The `time` variable contains the timestamp while the `vicon` variable contains the VICON data in the following format:

$$\begin{bmatrix} x & y & z & \text{roll} & \text{pitch} & \text{yaw} & v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^T$$

2 Grading

You will be evaluated based on the Root Mean Square (RMS) error between your estimated pose of the robot and the ground truth VICON data, specifically, the norm of position RMS errors in x, y, z and the norm of orientation RMS errors in q_w, q_x, q_y, q_z. Your `estimate_pose.m` implementation must also run in less than 1ms on average. In addition, you must properly handle edge cases such as when there are no IDs detected in the image or the senor data is invalid (refer to Section 1.3 and the header description for `estimate_pose.m` for more information). Further grading details including test thresholds can be found on canvas.

3 Submission

You will use the `turnin` system for submitting your code for this phase. The project name for this assignment is called `proj2phase1`, so the command for `turnin` would be

```
$ turnin -c meam620 -p proj2phase1 -v *
```

Your submission should include:

- A `README` file containing anything specific we should be aware of (including the names of any collaborators and a description of the collaboration).
- Files `init_script.m`, `estimate_pose.m` and any additional files you use for your computations.

Shortly after submitting using `turnin` you should receive an email from `meam620@seas.upenn.edu` stating that your submission has been received. After that, you can check the status of your submission on the monitor webpage at https://alliance.seas.upenn.edu/~meam620/monitor/.

Once the automated tests finish for your code, you should receive an email containing your test results. This email will contain plots of your estimates compared to the ground truth and also tell you how long

each iteration of your code takes. You should write your own functions for evaluating the error between your estimate using vision and the ground truth. Your grade would depend on the time taken per iteration and the error between your estimate compared to ground truth.

References

[1] AprilTags, http://april.eecs.umich.edu/wiki/index.php/AprilTags