# Progress Review 10

Cole Gulino

Team C / Column Robotics

Teammates: Job Bedford, Erik Sjoberg, Rohan Thakker

ILR # 9

March 17, 2016

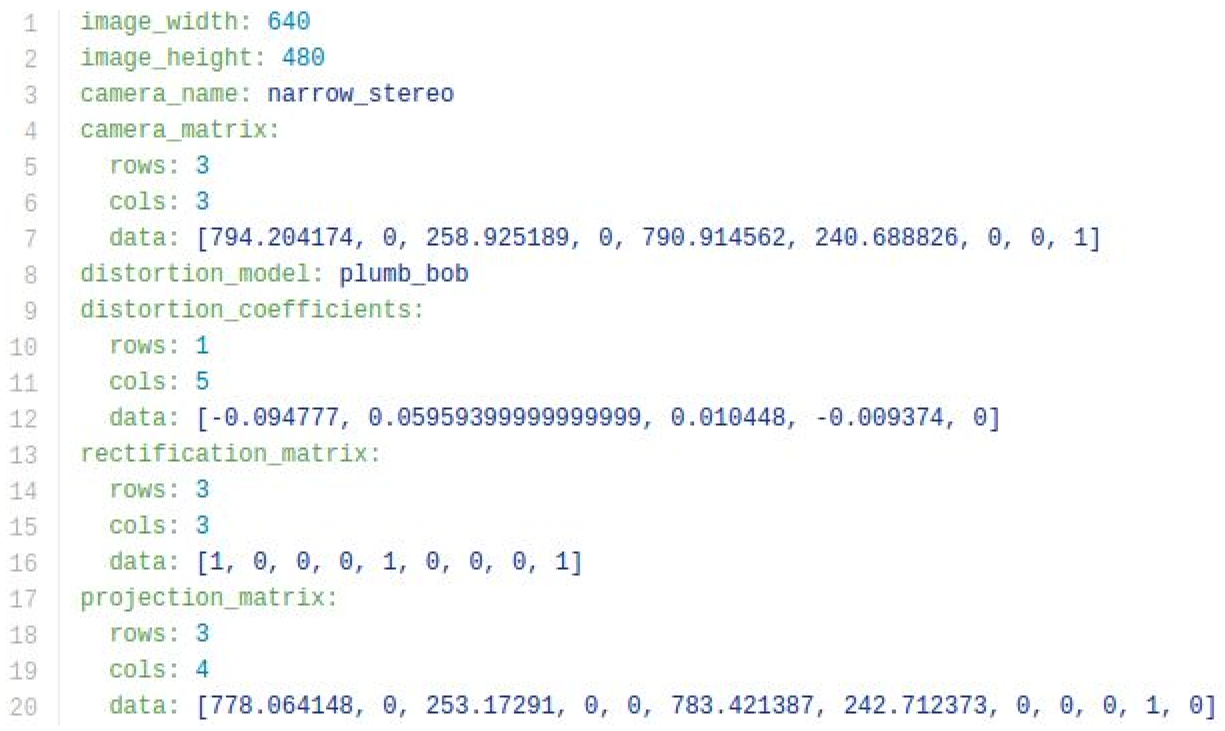
## Individual Progress

Since the last PR I have focused my efforts on getting the April Tag updates to be more effective.

In order to ensure the effectiveness of the April Tags, I needed to make sure that the camera was properly calibrated and that the position estimates that we get from the April Tag are invariant to the rotation of the quadcopter.

The first thing that I did was to calibrate the camera using the ROS package `camera_calibration` [1].

The calibration `.yml` file is shown in Figure 1:



```
image_width: 640
image_height: 480
camera_name: narrow_stereo
camera_matrix:
  rows: 3
  cols: 3
  data: [794.204174, 0, 258.925189, 0, 790.914562, 240.688826, 0, 0, 1]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.094777, 0.05959399999999999, 0.010448, -0.009374, 0]
rectification_matrix:
  rows: 3
  cols: 3
  data: [1, 0, 0, 0, 1, 0, 0, 0, 1]
projection_matrix:
  rows: 3
  cols: 4
  data: [778.064148, 0, 253.17291, 0, 0, 783.421387, 242.712373, 0, 0, 0, 1, 0]
```

Figure 1) .yml File for the Playstation Eye Camera

I included this file in the `.ros/camera_info` folder to be used by gscam. I also updated the focal length by getting the [0][0] and [1][2] elements of the camera intrinsic matrix. I then updated the camera focal length in x and y in the April Tag publishing node.

This gave us better results and more accuracy in the pose estimation provided from the April Tag.

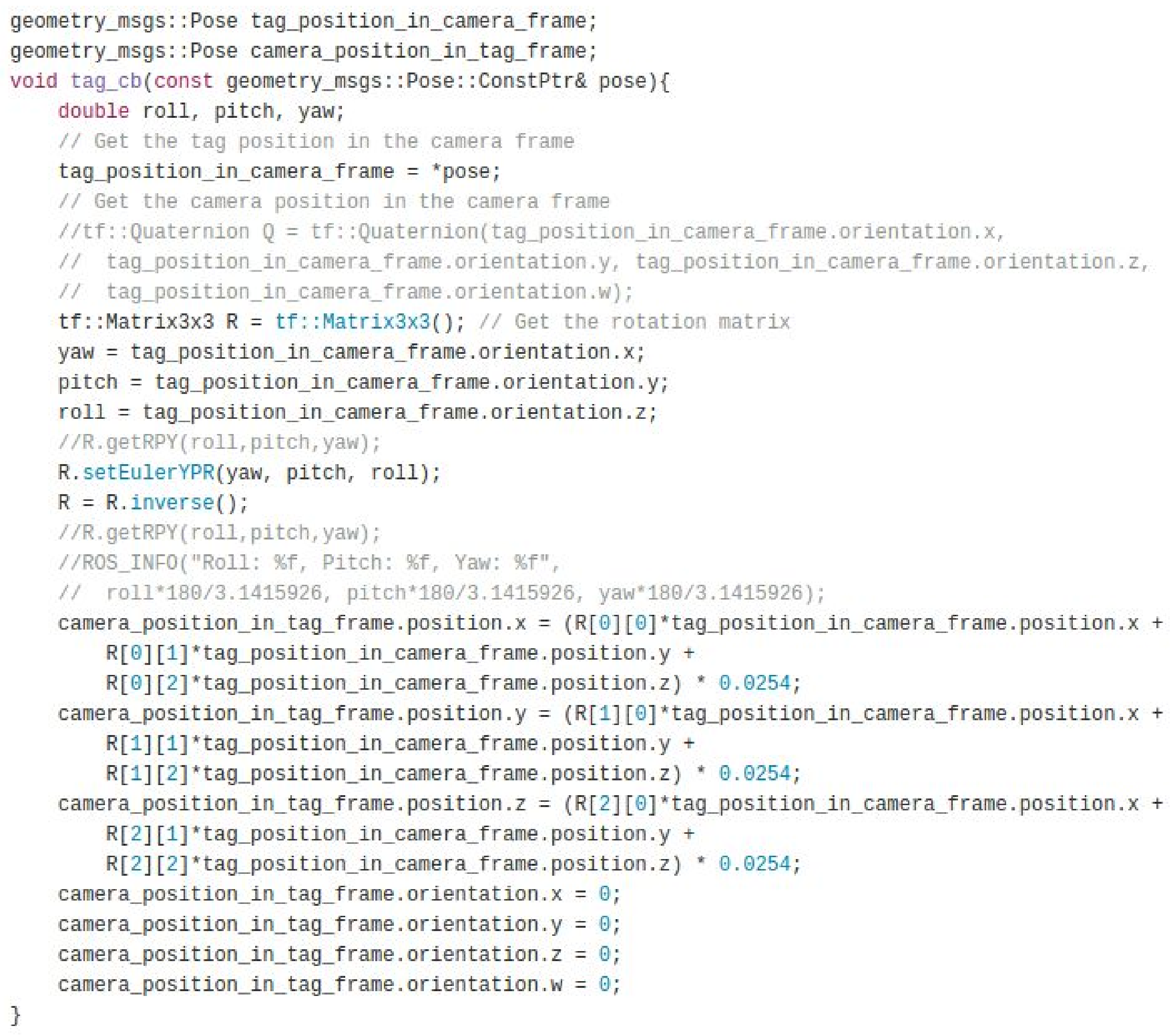
The next thing that I did was to manipulate the position estimates in order to get estimates that were independent of the orientation of the quadcopter.

Out of the box, the April Tag node provides the April Tag's pose in the camera's frame. As the quadcopter rotates, the coordinate frame changes as the x-y-z direction will flip as the quadcopter rotates. We would like to have a way to have a consistent global frame. In order to do this, we would like to get the camera's pose in the April Tag's frame.

In order to get the camera's pose in the April Tag's frame, we need to transform the coordinate frame. We can do this by getting the orientation of the quadcopter from the April Tag updates. These provide us with Euler Angles which we can convert to a rotation matrix. And then we perform the following matrix multiplication:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{camera}} = R_{\text{camera}}^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{april tag}}$$

This gave us the rotationally invariant updates that we need to have a consistent global frame. Figure 2 shows the code that I wrote to provide the updates a callback function for the April Tag node updates:



```
geometry_msgs::Pose tag_position_in_camera_frame;
geometry_msgs::Pose camera_position_in_tag_frame;
void tag_cb(const geometry_msgs::Pose::ConstPtr& pose){
    double roll, pitch, yaw;
    // Get the tag position in the camera frame
    tag_position_in_camera_frame = *pose;
    // Get the camera position in the camera frame
    //tf::Quaternion Q = tf::Quaternion(tag_position_in_camera_frame.orientation.x,
    //  tag_position_in_camera_frame.orientation.y, tag_position_in_camera_frame.orientation.z,
    //  tag_position_in_camera_frame.orientation.w);
    tf::Matrix3x3 R = tf::Matrix3x3(); // Get the rotation matrix
    yaw = tag_position_in_camera_frame.orientation.x;
    pitch = tag_position_in_camera_frame.orientation.y;
    roll = tag_position_in_camera_frame.orientation.z;
    //R.getRPY(roll,pitch,yaw);
    R.setEulerYPR(yaw, pitch, roll);
    R = R.inverse();
    //R.getRPY(roll,pitch,yaw);
    //ROS_INFO("Roll: %f, Pitch: %f, Yaw: %f",
    //  roll*180/3.1415926, pitch*180/3.1415926, yaw*180/3.1415926);
    camera_position_in_tag_frame.position.x = (R[0][0]*tag_position_in_camera_frame.position.x +
        R[0][1]*tag_position_in_camera_frame.position.y +
        R[0][2]*tag_position_in_camera_frame.position.z) * 0.0254;
    camera_position_in_tag_frame.position.y = (R[1][0]*tag_position_in_camera_frame.position.x +
        R[1][1]*tag_position_in_camera_frame.position.y +
        R[1][2]*tag_position_in_camera_frame.position.z) * 0.0254;
    camera_position_in_tag_frame.position.z = (R[2][0]*tag_position_in_camera_frame.position.x +
        R[2][1]*tag_position_in_camera_frame.position.y +
        R[2][2]*tag_position_in_camera_frame.position.z) * 0.0254;
    camera_position_in_tag_frame.orientation.x = 0;
    camera_position_in_tag_frame.orientation.y = 0;
    camera_position_in_tag_frame.orientation.z = 0;
    camera_position_in_tag_frame.orientation.w = 0;
}
```

Figure 2) Callback Function from April Tag Node

The orientation of the April Tag is a left-handed frame unfortunately and is specified as in Figure 3:

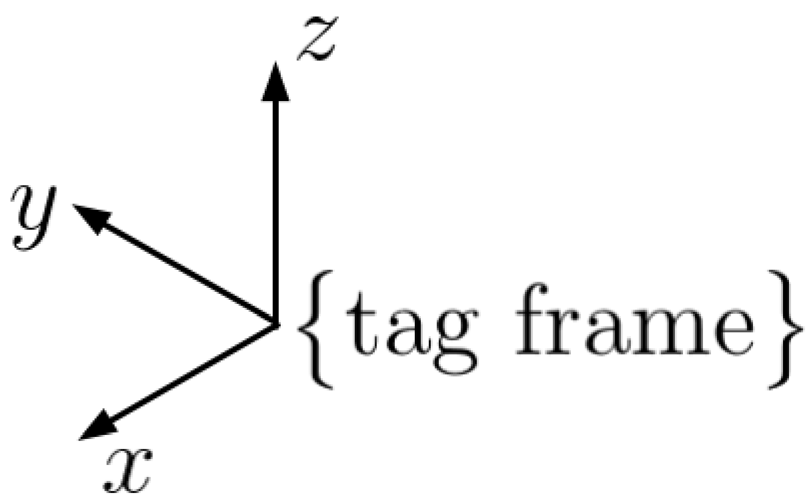


Figure 3) Coordinate Frame of the April Tag

## Challenges

The major challenges that we faced during this PR was similar to the major challenges that we have been facing ever since we started developing software for the quadcopter. Improper knowledge about the coordinate frames is a very challenging problem. The April Tag, body, and local frames are all inconsistent.

This has required a lot of experimentation which is not always consistent over trials. We still have not fully locked down all of the coordinates, but we have most of them written down and documented.

Another major challenge that we have been facing is the slow develop-build-test pipeline. We have automated this with a script, but we often need to reboot the quadcopter after every run. This requires us to SSH into the quad and run the script which can take up to 5 minutes if the connection is poor.

We were not able to get as much precision as we would like to have had on our landing, but we have successfully gotten April Tag updates that are stable. They are changing in the way that we would expect, and we feel very confident in them. The next step is for us to get the body frame and the velocity update frames locked down. There is some idea that the odometry body frame is different than the frame we use to publish the setpoint velocities of the quadcopter. Once we have this down, we believe that we will have very stable position locking on the quadcopter. We have found that we have accurate locking in at least one direction, which gives us much optimism.

## Teamwork

I did not do much in the team setting this PR. I focused mainly on solo work as I was out of town for Spring Break and sick for the tail end of it.

The rest of the team worked together on testing the April Tag node I wrote along with their P control loop. We needed the P control loop, because the internal control loop of the Iris+ is much too aggressive for the small distances around the April Tag.

Job and Rohan worked on the P control before Spring Break, and Erik tested the P control node with my April Tag node after he got back from out of town.

We all worked together to document and use version control with github [2] [3]. This system has worked well so that we can keep track of contributions and keep our versions safe in a remote repository.

## Future Work

By the next PR, we are going to be continuing to work on closed-loop April Tag servoing and landing while branching out to working on Kalman Filters and cone search.

Erik will be working on the autonomous cone search now that the large net is completed. This will allow us to lock down how to control the quadcopter using its built-in coordinate frames.

Job and I will be working on getting the P controller to work for precision landing similar to the last demonstration but with higher accuracy.

Rohan will be digging into the internal Kalman Filter of the Pixhawk flight controller in order to update the global frame with the April Tag updates. This is work that is parallel to what Job and I are doing. By updating the odometry (global) frame (which floats due to measurement error), we should be able to move around in the world frame without needing to update our position based on the relative position of the quadrotor to the April Tag.

## Resources

[1] ROS camera_calibration package

[2] Column Repository

[3] april_tag Repository