Final Report

16-667 Robot Autonomy

Quadrotor State Estimation
and Obstacle Detection
May 5 - Spring 2016

Job Bedford, Cole Gulino, Erik Sjoberg and Rohan Thakker

Problem Statement

Implement state estimation and obstacle detection to enable a quadrotor to navigate in indoor (GPS denied) environment without crashing into obstacles.

Objectives

1. Simulate the dynamics of the quadrotor
2. Literature survey on planning for quadrotors and differential flatness
3. Implement a cascaded controller to control the position and heading of the quadrotor
4. Implement a Kalman Filter in simulation
5. Implement state estimation and obstacle detection by sensor fusion of data from IMU, downward facing camera and forward facing RGB-D sensor

Dynamics

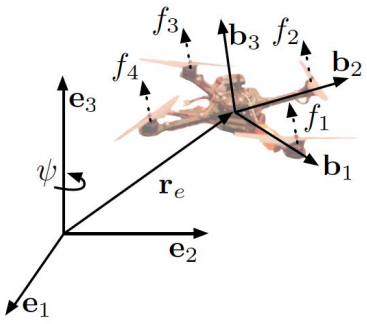

Figure 1) Frame convention for quadrotor

As shown in figure 1, [e1, e2, e3] and [b1,b2,b3] represent the [x,y,z] vectors of the world and body frame.

Control inputs:

$$[f\ \mathrm{M}]^T, f \in \mathbb{R}, \mathrm{M} \in \mathbb{R}^3$$

State:

$$[x\ v\ R\ \Omega]^T$$

Dynamics:

$$\dot{x} = v$$
$$m\dot{v} = mge_3 - fRe_3$$
$$\dot{R} = R\hat{\Omega}$$
$$J\dot{\Omega} + \Omega \times J\Omega = \mathrm{M}$$

As shown above, we are using a 18x1 state vector.
x = position of the body frame of the quadrotor in the world
v = velocity of the body frame of the quadrotor in the world
R = Rotation matrix of the body frame represented in the world frame
Omega = Angular velocity of the body frame of the quadrotor
J = moment of inertia
M = mass
f = thrust vector of the quadrotor
M = Moment vector of the quadrotor

Differential Flatness and Planning

The paper that we reference [1] shows that the system is differentially flat if there exists a set of flat outputs, Such that the state and the control inputs can be represented as a function of these flat outputs and their derivatives.

$$y = y(x, u, \dot{u}, \ldots, u^{(p)}) \qquad \begin{aligned} x &= x(y, \dot{y}, \ldots, y^{(q)}) \\ u &= u(y, \dot{y}, \ldots, y^{(q)}). \end{aligned}$$

We can show the any 4 out of these 6 variables [x,y,z,roll, pitch, yaw], can serve as flat outputs for the quadrotor. Hence, we can directly get dynamically feasible trajectories using any smooth trajectories in flat space. [2] uses polynomials in the C_free space and finds its coefficients by minimizing the snap of the trajectories because moment is a function of the snap of the position.

Control Architecture

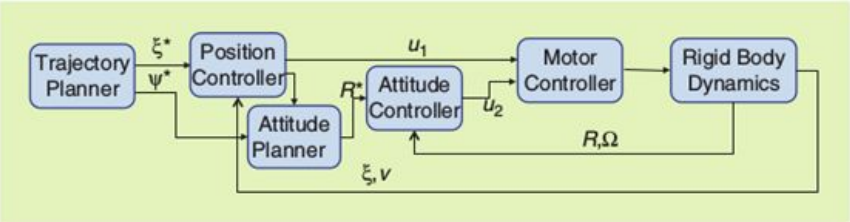

Figure 2) Control Architecture for the Quadrotor

Figure 2 shows the control architecture that we used on the robot. The trajectory planner publishes desired position [x,y,z] and heading [yaw] of the quadrotor to the position controller. The position controller runs a PID to calculate the desired roll, pitch, yaw (we use a rotation matrix to represent this). Attitude planner produces a smooth trajectory to this orientation and passes it to the attitude controller which run a PID loop to produce the desired thrust on the motors.. Note that this is a cascaded control architecture. Hence, the frequency at which the attitude controller runs is more than 10 times faster than the frequency at which we run the position controller.

State Estimation: Extended Kalman Filter

The planning and control system discussed up to this point has been based on the assumption that we completely know our state. On a real robot, this is obviously not the case. There is considerable uncertainty due to sensor noise and error. In order to filter out the noisy sensor information, we use an Extended Kalman Filter.

The Extended Kalman Filter can fuse multiple data sources while still providing a state estimation during lapses in sensor information through the prediction phase.

We have designed and implemented a 3D simulator for arbitrary quadcopter dynamics. We have also implemented an API in MATLAB for a Kalman Filter, Robots and Sensors. This code is extendable to any model of quadcopter dynamics that we give it. It can also be used for any arbitrary sensor.

We have profiled and simulated a velocity sensor which is indicative of the current optical flow sensor that we are using on the quadcopter. The dynamics that we are using is currently a point mass model that we assume has no control inputs. Figure 4 shows how the state estimation performs with gaussian white noise in the process and measurements from the velocity sensor. Ground truth is a dashed line, while the estimate is a solid line. As you can see, the estimate in position has a random walk associated with integrating gaussian white noise of a derivative measurement.

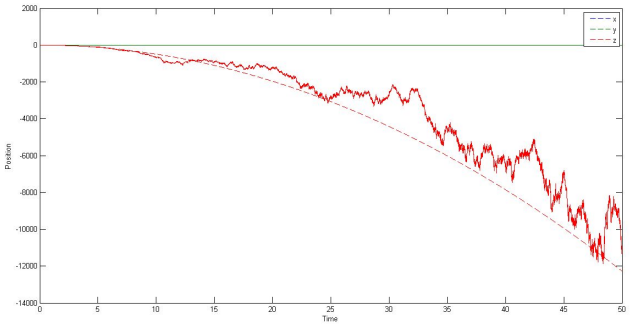

Figure 4) Linear Drift Associated with Integrating Velocity Measurements

This simulated environment is highly indicative of our overall system as explained in the following State Estimation: Optical Flow section.

State Estimation: Optical Flow

As mentioned in the previous section. Our hardware system for state estimation can be profiled as an Extended Kalman Filter that integrates velocity information from an optical flow camera and acceleration updates from an IMU. The IMU comes integrated directly with the system and the estimates are directly passed onto the onboard EKF. We used the PX4FLOW [2] camera for optical flow. This sensor can attach directly to the IMU which can be configured to provide velocity updates to the Extended Kalman Filter.

The PX4FLOW camera calculates the relative motion between the camera and projected pixel coordinates given by the equation.

$$\mathbf{V} = -\mathbf{T} - \omega \times \mathbf{P}$$

Where $\mathbf{P}$ is the 3D coordinates of the projected pixel coordinates, $\mathbf{T}$ is the relative translation, and ω is the angular velocity. The paper expresses the idea of "flow" and velocity as.

$$v_x = \frac{T_z x - T_x \mathrm{f}}{Z} - \omega_y \mathrm{f} + \omega_z y + \frac{\omega_x xy - \omega_y x^2}{\mathrm{f}}$$

$$v_y = \frac{T_z y - T_y \mathrm{f}}{Z} + \omega_x \mathrm{f} - \omega_z x + \frac{\omega_x y^2 - \omega_y xy}{\mathrm{f}}$$

Figure 5 shows the raw velocity estimates and the position estimates from the EKF in the configuration where the drone is maintaining position.

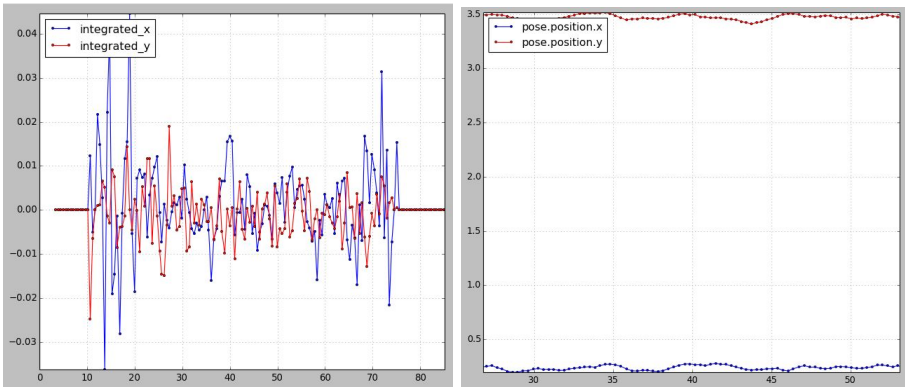

Figure 5) (Left) Raw Velocity Measurements. (Right) Position Estimates from EKF

These estimates show a strong ability to maintain position. There is only a slight deviation in x and y which can be attributed to small perturbations in the environment or small errors in the controller.

In the State Estimation: Extended Kalman Filter section, we discussed the linear drift for integrating velocity estimates in the EKF. Figure 6 shows the linear drift using on the live system.

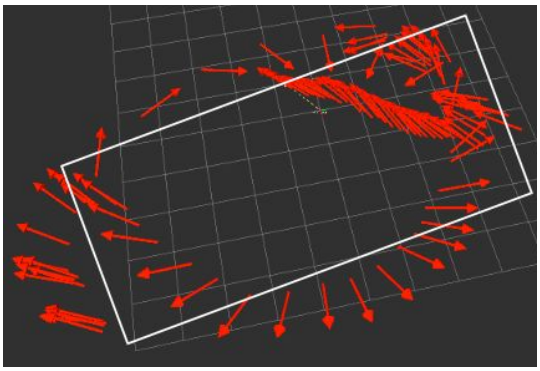

Figure 6) Odometry Information vs. Ground Truth

Figure 6 shows odometry information vs. ground truth. The base of the arrows represent a pose in 3D space and the arrow direction and length is the velocity at the current time. As can be seen from the Figure, the top left corner of the box is where the system starts, as can be seen there is significant drift as the quadrotor moves.

SLAM

For the depth sensor we used an Asus Xtion Pro Live [3]. This camera is an RGB-D sensor that uses structured light in order to improve its 3D stereo estimates. The mapping package that we used is called RTAB-Map [4]. This package is a graph and node based SLAM system with a highly maintained ROS wrapper. The package uses TORO graph optimization techniques and uses bag-of-words model in order to detect loop closures in the images. We used the OpenNI package in order to handle the point cloud information.

Figure 7 shows the 3D map while standing still and while moving around the quadrotor by hand.

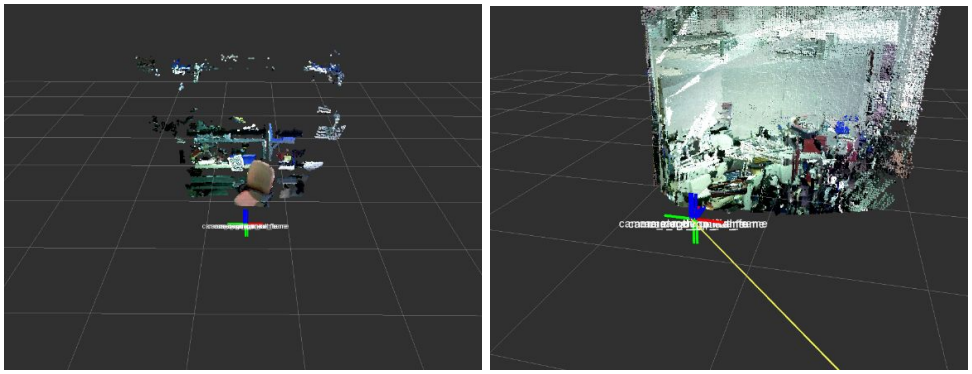

Figure 7) Mapping Information for Stationary (Left) and Moving by Hand (Right)

This information was quite stable when stationary, but whenever the quadrotor was moving, the SIFT features detected by the algorithm was not able to generate enough inliers to determine the transformation between the two frames. This made it ineffective to use as this would cause the odometry information to fail. We were only able to get information at around 0.5 Hz. This was too slow for our needs.

Obstacle Detection

Even though the SLAM information was too computationally intensive to run on our onboard computer, we wanted to be able to detect and avoid obstacles. We were able to get the point cloud data at around 20 Hz. This was more than fast enough for our needs. We took the point cloud information and projected it down into the 2D plane. Figure 8 shows detected obstacles projected down into a 2D cost map. The points are filled and cleared using ray-tracing on the 3D point cloud information. The cost map is a local map that only persists for as long as the quadcopter is within some range of the points.

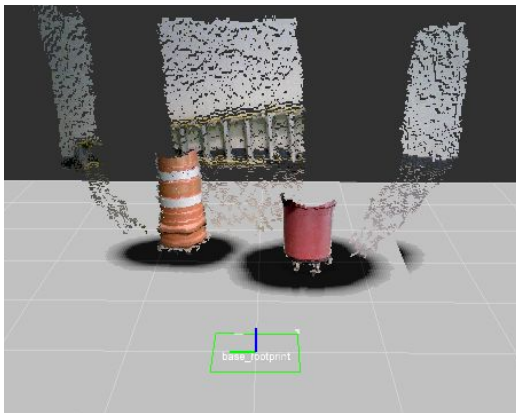

Figure 8) Point Cloud Projected onto a 2D Cost Map

By getting the point cloud data at around 20 Hz, we were able to fill and clear the cost map at around 3 Hz comfortably, which is an effective speed for our control and planning algorithms.

Conclusions

State estimation is one of the most fundamental technological challenges of the quadrotor system. The small computational power of on board computers and the constant motion of a quadrotor in flight make it incredibly difficult to estimate the state of the quadrotor effectively indoors where the quadrotor is GPS denied. SLAM techniques for estimation pose is an effective way to make up for the drift associated with integrating velocity updates into an EKF, but the algorithms are in general too computationally expensive. By planning locally using fast techniques such as optical flow in an environment with good features along with low-weight local mapping techniques, one can effectively plan and navigate a quadrotor in an indoor environment without the need for a global position estimate.

We have a final video that showcases our system in its current state. The video has slides describing various parts of the project with videos after them to show our progress. The video can be seen at the following link: https://youtu.be/lohw-BtURTQ.

References

[1] Murray, Richard M., Muruhan Rathinam, and Willem Sluis. "Differential flatness of mechanical control systems: A catalog of prototype systems."ASME international mechanical engineering congress and exposition. 1995.
[2] Mellinger, Daniel, Nathan Michael, and Vijay Kumar. "Trajectory generation and control for precise aggressive maneuvers with quadrotors." The International Journal of Robotics Research (2012): 0278364911434236.
[3] Dominik Honegger, Lorenz Meier, Petri Tanskanen and Marc Pollefeys. “An Open Source and Open Hardware Embedded Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications”, ICRA 2013
[4] Asus Xtion Pro Live
[5] M. Labb. Online global loop closure detection for large-scale multi-session graph-based slam. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, Sept 2014.
[6] Mahony, Robert, Vijay Kumar, and Peter Corke. "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor." IEEE Robotics & amp amp Automation Magazine 19 (2012): 20-32.