Fault-Tolerant Mapping and Localization for Quadrotor UAVs

Maximillian Gilson, Jason Gauthier, and Xiaodong Zhang
Wright State University, Dayton, OH, 45435, USA

Kaylee Garcia and Cory Kienzle
California Baptist University, Riverside, CA 92504, USA

Jonathan Muse
Air Force Research Laboratory, Dayton OH, 45433, USA

This paper presents a fault-tolerant control method for a quadrotor UAV using solely on-board sensors. A simultaneous localization and mapping (SLAM) system is developed utilizing a laser rangefinder and an open source SLAM algorithm called GMapping. This system allows for mapping of the surrounding environment as well as localizing the position of the quadrotor, enabling real-time position control. However, the SLAM system using the laser rangefinder may fail in certain degenerate environment like featureless tunnels or straight hallways. In order to compensate for possible faults in the SLAM measurements, a fault detection and fault-tolerant control method is developed. An observer is designed to estimate the translational velocity of the quadrotor using SLAM position measurements. The fault detection residual is defined as the deviation between this SLAM-based velocity estimate and another velocity estimate generated by an optical flow algorithm utilizing measurements provided by a downward facing camera. Real-time experimental results have shown the effectiveness of the fault-tolerant control algorithm.

I. Introduction

In recent decades, unmanned aerial vehicles (UAVs) have attracted significantly increasing attention because of their wide variety of potential applications. Quadrotor is a specific type of UAV where there are four propellers in an "X" or "+" configuration. This type of UAV allows for vertical take off and land (VTOL), hovering, and a small form factor when compared to other types of UAVs.

Research has shown interesting results on indoor mapping capabilities for quadrotor UAVs [1]. A fully autonomous indoor quadrotor with a three-dimensional simultaneous localization and mapping (SLAM) system was shown in [2]. SLAM systems often use a laser-rangefinder approach, a forward facing depth-sensing camera, or vision-based systems [3]. Among these methods, laser-rangefinder-based SLAM systems have proven to be accurate and robust in most scenarios. However, it has been observed that featureless environments like tunnels or straight hallways will cause position faults in SLAM, which may lead to instability of the flight control system or even crash [4]. Therefore, in order to ensure reliable and safe operations of UAVs, there is a significant need in the development of fault detection and accommodation schemes for laser-rangefinder-based SLAM systems.

This paper focuses on the development of a fault-tolerant control system for a quadrotor UAV using solely on-board sensors. First, a SLAM system using a laser rangefinder and an open source SLAM algorithm called GMapping is implemented. Second, a fault detection scheme is designed to detect possible faults in the SLAM position measurements. Additionally, a fault-tolerant control scheme is developed to accommodate SLAM faults and stabilize the quadrotor. Third, experimental results using a real-time quadrotor test environment have shown the effectiveness of the proposed fault-tolerant control scheme.

II. Quadrotor Dynamic Model

The dynamic quadrotor model used in this paper consider the gravity, the thrust and moments generated by the rotors, and drag forces acting on the quadrotor body. Let us define the following twelve state variables:

$$[p_{xi}\ p_{yi}\ p_{zi}\ u_i\ v_i\ w_i\ \phi\ \theta\ \psi\ p\ q\ r]^T$$

where $[p_{xi},\ p_{yi}, p_{zi}]^T$ and $[u_i, v_i, w_i\ ]^T$ represent the position and velocity of the quadrotor in the inertial frame, respectively, $[\phi, \theta, \psi\ ]^T$ are the roll, pitch and yaw angles of the quadrotor, respectively, and $[p, q, r\ ]^T$ are the angular rates of the quadrotor. Thus, by using the Newton-Euler equations of motion, a state space model of the quadrotor can be represented as follows [5]:

$$\begin{bmatrix}\dot{p}_{xi}\\ \dot{p}_{yi}\\ \dot{p}_{zi}\end{bmatrix} = \begin{bmatrix}u_i\\ v_i\\ w_i\end{bmatrix} \tag{1}$$

$$\begin{bmatrix}\dot{u}_i\\ \dot{v}_i\\ \dot{w}_i\end{bmatrix} = \frac{1}{m} R_b^e(\psi, \theta, \phi)\left(\begin{bmatrix}0\\ 0\\ -F\end{bmatrix} - c_d V_b\right) + \begin{bmatrix}0\\ 0\\ g\end{bmatrix} \tag{2}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = R_\rho(\phi, \theta)\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{3}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\left(\frac{I_y - I_z}{I_x}\right)pr\\ \left(\frac{I_z - I_x}{I_y}\right)pr\\ \left(\frac{I_x - I_y}{I_z}\right)pq\end{bmatrix} + \begin{bmatrix}\left(\frac{1}{I_x}\right)\tau_\phi\\ \left(\frac{1}{I_y}\right)\tau_\theta\\ \left(\frac{1}{I_z}\right)\tau_\psi\end{bmatrix} \tag{4}$$

where R_b^e represents the rotation matrix from the body frame to the inertial frame, F is the thrust force generated by the propellers, m is the mass of the quadrotor, g is acceleration from gravity, I_x, I_y, I_z, are the moments of inertia in the directions of body frame axes x, y, z, respectively, τ_ϕ, τ_θ, τ_ψ, are the roll, pitch, and yaw torques, the matrix R_ρ represents the transformation between angular rates and the Euler angle rates given by

$$R_\rho(\phi, \theta) = \begin{bmatrix}1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta)\\ 0 & \cos(\phi) & -\sin(\phi)\\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta)\end{bmatrix},$$

the rotation matrix R_b^e in Eq. (2) is defined based on a 3-2-1 rotation sequence as

$$R_b^e(\psi, \theta, \phi) = \begin{bmatrix}c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi\\ c\theta s\psi & s\phi s\theta s\psi - c\phi c\psi & c\phi s\theta s\phi - s\phi c\psi\\ -s\theta & s\phi c\theta & c\phi c\theta\end{bmatrix},$$

where $s\,\cdot$ and $c\,\cdot$ are shorthand notations for the $sin(\cdot)$ and $\cos(\ \cdot)$ functions, respectively. Additionally, the term $c_d V_b$ in Eq. (2) represents the drag force acting on the vehicle frame, with C_d being a constant drag force coefficient and V_b representing the translational velocity of the quadrotor relative to the body frame.

III. SLAM Using Laser Rangerfinder

The objective of a simultaneous localization and mapping algorithm is to generate a mapping of an agent's surrounding environment while simultaneously calculating the position and rotation of the agent within this environment. In this paper, the Rao-Blackwellized particle filtering (RBPF) based SLAM algorithm is used to generate maps and position estimates in two-dimensional space.

A. SLAM Algorithm Description

The RBPF method is an effective solution to the SLAM problem. The algorithm is briefly described below.

By using the observations $z_{1:t} = z_1, \cdots, z_t$ and the odometry measurements $u_{1:t} = u_1, \cdots u_{t-1}$ obtained by the agent, the primary function of RBPF in SLAM is to estimate the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the map m and the trajectories $x_{1:t} = x_1, \cdots, x_t$ of the agent based on the following factorization [6]:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t})p(x_{1:t} \mid z_{1:t}, u_{1:t-1})\ . \tag{5}$$

Eq. (5) allows us to first compute the robot's trajectory and then to generate the map based on the trajectory. Since the posterior over maps $p(m \mid x_{1:t}, z_{1:t})$ can be computed analytically using pose estimate of the agent, the algorithm can be calculated efficiently.

To estimate the posterior, RBPF has individual maps assigned to every sample. These maps are generated based on the observations and trajectory of each respective particle. The Rao-Blackwellized method utilizes a sampling importance resampling (SIR) filter that incrementally processes observations and odometry as soon as they become available. In this paper, the implementation of SLAM is based on GMapping [7]. GMapping uses RBPFs combined with an adaptive proposal and selective resampling technique to provide more efficient and accurate localization and map estimates. This process is summarized by the following steps [7]:

1) Next generation of particles are generated from the current generation by sampling the proposal distribution;
2) A weight based on the posterior and proposal distribution is assigned to each particle;
3) Resampling occurs which replaces low weighted particles with higher weighted particles;
4) Map estimates for each pose sample are calculated based on trajectory and history of observations.

In this research, the maps generated are in the form of occupancy grids. This is essentially a large matrix with each element having a numerical value. These numerical values show whether or not the space in the map is an obstacle, free space, or unknown space.

B. SLAM Algorithm Implementation

The SLAM algorithm was implemented using open-source packages from the robot operating system (ROS). Specifically, several packages from ROS were used in this research, including rosbridge, laser_scan_matcher, and gmapping. Additionally, an extended Kalman filter (EKF) is used to smooth the pose estimates [5]. A block diagram detailing the high-level architecture is shown in Figure 1. A brief description of each component is given below.

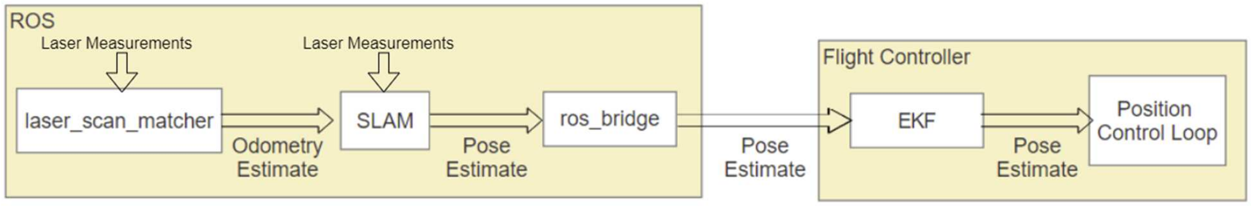

Figure 1 Block diagram for SLAM implementation

ROS is a software platform for open-source tools that allow users to easily control robotic systems and incorporate advanced functionality. The following three packages were utilized to solve the SLAM problem.

1) Rosbridge is a package that allows for TCP communication between ROS and another system. In this case the two systems being connected were ROS and the flight controller model, both running onboard the quadrotor. This allows ROS to send pose estimates to the flight controller, so that the quadrotor's position can be controlled.
2) The second package used in this research was laser_scan_matcher. This package allows for odometry estimates. As mentioned above, it is necessary to provide odometry updates to the SLAM algorithm. The laser_scan_matcher package functions by observing the surrounding environment from the laser rangefinder measurements and using this to estimate the current position. This package works independently of SLAM, which allows for a faster and more efficient pose calculation. Once the pose estimates are provided to the gmapping package, gmapping can generate a more accurate map.
3) Gmapping is the third and final package used. This package is an implementation of the aforementioned RBPF-based SLAM. Using this implementation, there are various parameters that can be selected. The main parameters focused on in this research were the number of particles and the resolution of the map. A total of 15 particles were chosen and a resolution of 0.3 meters/occupancy grid pixel was used. These parameters have the largest impact on computational performance and accuracy of the map. From this package the map that is generated can be exported in text format as an occupancy grid, and the pose estimates can be sent to the rosbridge package for use in the flight controller.

Under certain circumstances there may be delays or inaccuracies in the SLAM pose estimates. To correct this, an EKF was used (see Figure 1), which uses sensor fusion and state estimation to give a better estimate when compared to only using sensor measurements. This is beneficial to this research due to issues of computational intensity, communication delay, or a lack of features in the surrounding environment. These issues can cause the SLAM algorithm to have large jumps or delays in the pose estimates, which could cause a crash during flight.

Pose estimation occurs at two stages: SLAM pose estimates and the EKF pose estimates. First, laser_scan_matcher provides an odometry update to the SLAM algorithm. This is a very efficient and fast algorithm. It uses the current environment visible to the laser range finder to calculate the position and rotation of the quadrotor. The estimate from the laser_scan_matcher is provided to SLAM which takes this as an odometry estimate. SLAM then uses this odometry estimate and the map that has been generated over the entire course of the flight to add an offset if there are any errors detected. Given the entirety of the map it is possible to detect errors over long distances. The combination of the odometry estimate from laser_scan_matcher and the corrections made from SLAM comprises the SLAM pose estimate. Lastly, the pose estimate from SLAM is fed into the EKF-based state estimation algorithm. The EKF takes the pose estimate from SLAM and smooths out the inconsistencies when either data is lost for a short period of time or there is a measurement error. This allows for more robust pose estimates.

IV. Fault-Tolerant Control

In certain cases, when there are few features in the environment, the laser-rangefinder-based SLAM system described in Section III may have trouble localizing position. This happens because there are not enough features in the environment to show that the quadrotor is moving in a particular direction, for instance, in long uniform hallways or tunnels. As a result, the SLAM position measurement will not be updated even if the UAV is moving. When this occurs, the controller will not be able to control the UAV position, and the quadrotor may become unstable in this direction. To accommodate the possible faults in the SLAM position measurements, a fault detection and accommodation system is needed.

Figure 2 shows a configuration of the proposed fault-tolerant system. First, in addition to the SLAM algorithm using laser rangerfinder described in Section III, an optical flow algorithm using a downward-facing camera estimates the velocity of the drone in *x* and *y* directions, respectively. Second, a fault detection algorithm is developed to estimate the velocity of the drone in *x* and *y* directions based on the SLAM position measurements and compares the velocity estimates generated using optical flow and SLAM, respectively. If the error between these two velocity estimates exceeds a predefined threshold, a fault in the SLAM system is determined. Consequently, the position feedback signal for quadrotor position control is switched from the SLAM position to the optical flow position. The optical flow position estimate is generated by simply integrating the velocity estimate provided by optical flow with respect to time. Below, we detail each of these three components.

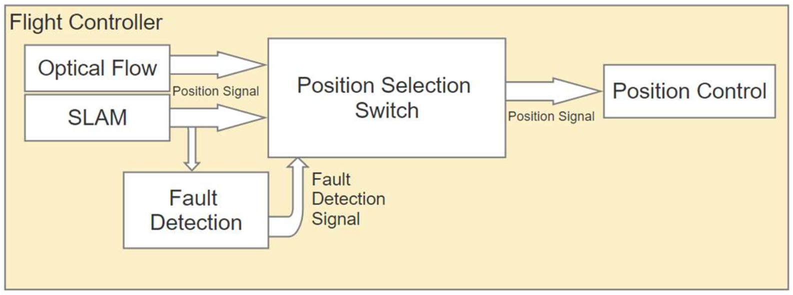

Figure 2 Configuration of the fault-tolerant system

A. Optical Flow

Optical flow is a method of approximating the motion of objects between successive images. In most cases the successive images are in the form of a video. There are many ways to calculate optical flow [8]. In this research, the motion of quadrotor in the x and y directions is estimated using a downward facing camera. The Lucas-Kanade method [9] for calculating optical flow functions is used in this research, which is briefly described below.

The Lucas-Kanade method assumes that the velocity of the pixels in the image are approximately constant. A system of optical flow equations is then solved by the least squares criterion. First a set of partial derivative equations are developed:

$$\begin{gathered} I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ \vdots \\ I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{gathered}$$

where I_x, I_y, and I_t, are partial derivatives with respect to position x, y and time respectively, $q_1, \dots, q_n$ represent the n pixels in the image, and V_x and V_y are velocities in x and y directions, respectively. The above equations can be put in matrix form as:

$$Av = b$$

where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \qquad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}.$$

Using the least squares principle, the velocity vector can be estimated by:

$$v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} = (A^T A)^{-1} A^T b = \begin{bmatrix} \Sigma_i I_x(q_i)^2 & \Sigma_i I_x(q_i) I_y(q_i) \\ \Sigma_i I_y(q_i) I_x(q_i) & \Sigma_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\Sigma_i I_x(q_i) I_t(q_i) \\ -\Sigma_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

where $i = 1, \dots, n$.

Using this method, the velocity of the quadrotor in *x* and *y* directions can be estimated. For the quadrotor used in this research, there is a camera that is pointed towards the floor, that is, in the negative *z* direction. Considering the camera is fixed to the body of the quadrotor, the velocity estimates are in the body frame. A rotation matrix R_b^e is used to transform the optical flow velocity estimates from the body frame to the inertial frame. As the drone changes its position, the velocity in this direction is estimated.

B. Fault Detection Method

For fault detection, two velocity signals are needed to construct the residual signal. The second velocity estimate is obtained by using an observer utilizing SLAM position measurements. Specifically, from Eq. (1) and Eq. (2), the translational dynamics in the x and y directions can be represented in the inertial frame as follows [5]:

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{xi} \\ \dot{\boldsymbol{p}}_{yi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{v}_i \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} \dot{\boldsymbol{u}}_i \\ \dot{\boldsymbol{v}}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{c\theta c\psi} & \boldsymbol{s\phi s\theta c\psi - c\phi s\psi} & \boldsymbol{c\phi s\theta c\psi + s\phi s\psi} \\ \boldsymbol{c\theta s\psi} & \boldsymbol{s\phi s\theta s\psi + c\phi c\psi} & \boldsymbol{c\phi s\theta s\psi - s\phi c\psi} \end{bmatrix} \boldsymbol{a_m} \tag{7}$$

where $a_m \triangleq \begin{bmatrix} a_x \; a_y \; a_z \end{bmatrix}^T$ is the accelerometer measurement. By using a more compact form for the above equations and by taking into account the effect of SLAM measurement faults, we obtain:

$$\begin{aligned} \dot{\zeta} &= A\zeta + h(\phi, \theta, \psi, a_m) \\ \xi &= C\zeta \end{aligned} \tag{8}$$

where $\zeta = \left[p_{xi,} \, p_{yi,} u_i, v_i\right]^T$ is the state vector, $\xi = \left[\, p_{xi}, \; p_{yi} \,\right]^{\mathrm{T}}$ is the position measurement provided by SLAM. Additionally, the matrices

$$A = \begin{bmatrix} 0_{2x2} & I_2 \\ 0_{2x2} & 0_{2x2} \end{bmatrix}, h(\phi, \theta, \psi, a_m) = \begin{bmatrix} 0_{2x1} \\ R_{EB2} a_m \end{bmatrix}, \; R_{EB2} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \end{bmatrix},$$

the matrix $C = [I_2 \quad 0_{2\times 2}]$ with I_2 being a 2×2 identity matrix, $0_{2\times 2}$ is a 2×2 matrix with all entries zero, and $0_{2x1} = [0 \quad 0]$.

Based on these equations, a fault detection observer can be chosen as follows:

$$\begin{aligned} \dot{\hat{\zeta}} &= A\hat{\zeta} + h(\phi, \theta, \psi, a_m) + L_0(\xi - \hat{\xi}) \\ \hat{\xi} &= C\hat{\zeta} \end{aligned} \tag{9}$$

where $\hat{\zeta}$ and $\hat{\xi}$ are the estimated state vector and output vector, respectively, L_0 is the observer gain designed to make $\bar{A} = A - L_0 C$ asymptotically stable. Let us define the state estimation error as $\tilde{\zeta} = \zeta - \hat{\zeta}$ and the quadrotor position estimation error as $\tilde{\xi} = \xi - \hat{\xi}$. It follows that

$$\begin{aligned} \dot{\tilde{\zeta}} &= \bar{A}\,\tilde{\zeta} \\ \tilde{\xi} &= C\,\tilde{\zeta}\,. \end{aligned} \tag{10}$$

Therefore, in the absence of faults in the SLAM position measurement, the state estimation error $\tilde{\zeta}$ converges to zero.

By constructing the observer given by Eq. (9), the translational velocity can be estimated using the SLAM position measurements. However, if there is a position measurement loss from SLAM, the velocity estimate will be corrupted accordingly. This velocity estimate can then be compared to another velocity estimate provided by optical flow. If there is a large discrepancy between these two estimates, a fault in the SLAM position measurement is concluded.

Based on these two methods for estimating the translational velocity using different measurement sources, a fault detection scheme can be developed. Specifically, if there is a measurement loss in the x or y direction from SLAM, the velocity estimate in that direction generated by the observer will erroneous as well. While the quadrotor is still moving, the velocity estimate from the optical flow method will be significantly different from the SLAM velocity estimate. Thus, the faulty SLAM measurements can possibly be determined.

Therefore, for the purpose of fault detection, a residual signal is defined as the deviation between the velocity estimate using SLAM position measurement and the optical flow velocity estimate. The upper and lower thresholds on the residual signal chosen in this research are pre-defined through experiments by analyzing the residual signal in the absence of faults. The residual should remain within the thresholds in the absence of faults. If at a certain time, the residual exceeds the threshold, a fault in the SLAM measurements is detected. Once the fault is detected, a fault-tolerant control method for mitigating this fault will be activated.

C. Fault-Tolerant Control

The objective of the fault-tolerant control method is to mitigate the effect of the faulty SLAM position measurements, preventing unstable behaviors of the UAV. Under normal conditions, the quadrotor controls its position using the SLAM position measurements as a feedback signal. When a fault in the positional measurement occurs, the quadrotor will begin to accelerate along this axis, because the control system is unaware that the position is changing. Within a short period of time, the quadrotor can possibly collide with objects in the environment or crash. To avoid this situation, the faulty SLAM position signal must be switched to a more reliable source. This will help keep the quadrotor position feedback signal up-to-date and prevent a crash.

To calculate the positional updates from optical flow, the velocity estimates are simply integrated with respect to time. Switching from SLAM to optical flow allows for the quadrotor to maintain its stability and acceptable position tracking performance before the SLAM signal becomes available again. If certain features in the environment become available during flight, the SLAM position signal will be recovered. Specifically, if no fault is detected over a short period of time, the feedback signal for position control is switched back to SLAM.

To prevent issues with large jumps during switching, the initial conditions are appropriately chosen. When a fault is detected, the current value of the SLAM position is used for the initial condition for the optical flow localization. When the SLAM signal recovers and is no longer faulty, the position is switched back to the SLAM measurements, which is initialized by using the optical flow position. This method for switching allows for smooth transitions between the SLAM position and the optical flow position.

V. Experimental Results

A. Experimental Setup

The experimental setup is constructed of three parts, the quadrotor, Vicon motion capture system, and ground station computer. The Vicon system is mainly used to evaluate the effectiveness of the SLAM system and not needed in algorithm implementation. The quadrotor is comprised of six components, the embedded computer module, cameras, a laser range finder, a propulsion system, frame, battery, and localization markers. The embedded compute module is made up of two circuit boards, an Intel Aero compute board, which serves as the main computer for the quadrotor, and an expansion board which allows for easy access of I/O ports as well as provides a regulated power supply, programmable LEDs, and a switch to manually enable or disable the motors [10]. The Intel Aero board is a computer utilizing an Intel Atom processor running at 2.56 GHz with 4 GB of RAM and 32 GB of internal storage. It allows for connection to WiFi networks, and houses a variety of onboard sensors such as a 3-axis accelerometer and gyroscope, 3-axis magnetometer, temperature sensor, and pressure sensor.

The embedded computer module also connects to the cameras. There are two cameras on this quadrotor. One is in the direction of the x axis of the body frame, and the other is in the direction of the z axis of the body frame. The camera in the x direction, or the “front facing camera”, is not used in this research. The camera in the z direction is an OV7251 model 640x480 resolution VGA camera utilized in this research for optical flow based localization [10].

The laser rangefinder is also connected to the embedded computer module. It is a Hokuyo model URG-04LX-UG01. This laser rangefinder functions by supplying 682 measurements of position around a 240° field of view. These measurements are updated at 100 Hz. These measurements can be used to create a high resolution two-dimensional representation of the environment in which it exists.

The propulsion system is made up of four 2100 KV (RPMs/volt) brushless motors and 6-inch polycarbonate propellers. These motors are controlled by an electronic speed controller (ESC) which can be controlled via the Intel Aero computer board. The propellers are mounted on the motors, and both the four motors and ESC are mounted onto the quadrotor frame. The frame is a lightweight carbon fiber frame designed to resist impacts and be lightweight. The frame also houses all addition components such as the embedded compute module, cameras, and battery. The battery is a 3300 mAh Lithium-Ion rechargeable battery which powers all onboard components of the quadrotor.

The localization markers are related to the Vicon motion capture system. These localization markers are necessary for the Vicon system to calculate the position of the quadrotor in the inertial frame. They are reflective markers attached to the drone in a non-specific orientation which allow the four Vicon cameras to detect their position and accordingly triangulate their respective positions. The Vicon system is able to relay the position and angular calculations at a rate of 100 Hz with accuracy within a few millimeters.

The ground station computer uses Matlab and Simulink to implement the control algorithms and primary interfacing with the quadrotor. The control algorithm model is cross-compiled by Simulink to the quadrotor embedded computer module. This allows the drone to track a refined trajectory and the pilot to visualize data and information about the drone in real-time during flights. A photo showing the quadrotor with visible features is shown in Figure 3.

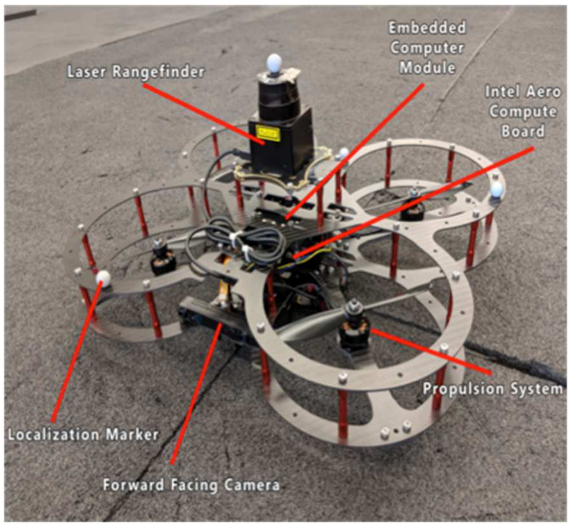

Figure 3 The quadrotor used in experiments with visible features denoted

B. SLAM Results

As mentioned in the previous section, the Vicon motion capture system allows for measuring the true position and rotation of the quadrotor. Using this truth value, the accuracy of SLAM can be evaluated. A circular trajectory flight experiment was conducted to show how accurate the SLAM pose estimates are during flight (shown in Figure 4). Moreover, the error of SLAM was plotted. This is the SLAM position measurements subtracted from the true positions provided by Vicon cameras. It was seen that the error over the entire course of the flight never exceeded 0.06 meters, which is considered reasonably accurate, as shown in Figure 5.

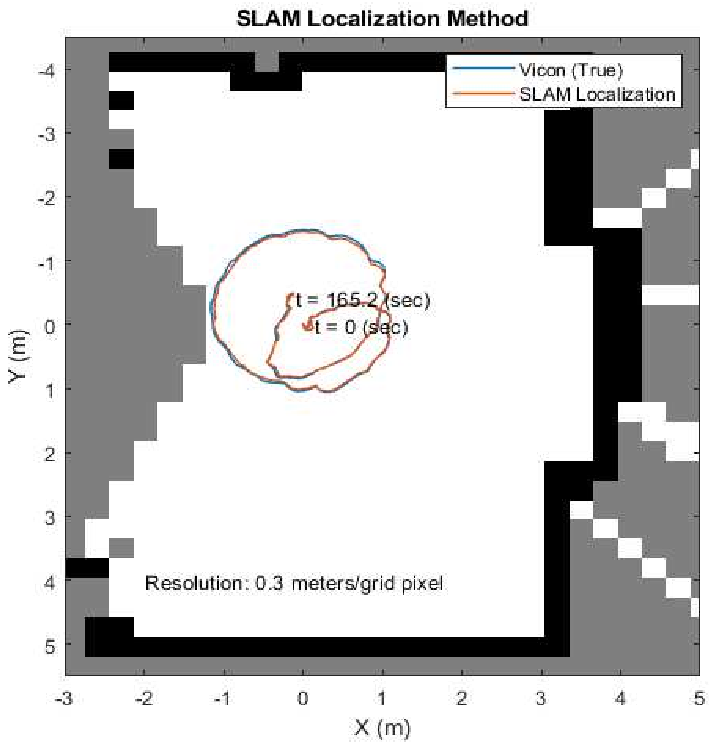

Figure 4 Circular flight showing SLAM and true position

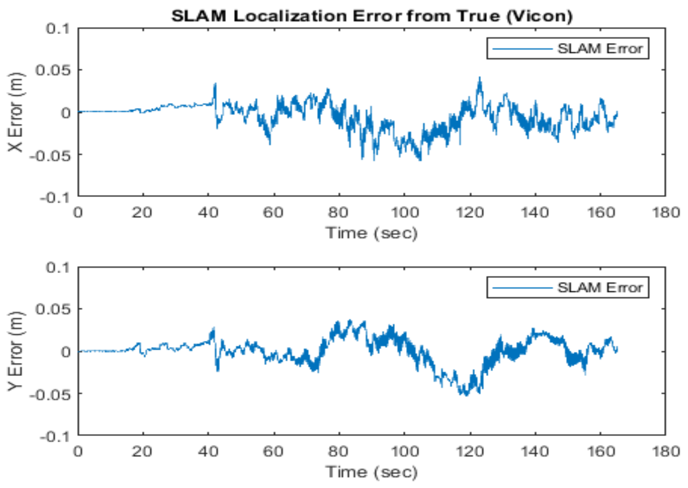

Figure 5 Localization error of SLAM in X and Y directions

C. Fault Detection

To illustrate the effectiveness of the fault-tolerant control method, a circular flight with the quadrotor was conducted. After flying for some duration, a fault in the SLAM measurements is injected to simulate faults in the SLAM position measurements in a featureless environment. Specifically, in the results presented below, the fault is injected by pausing the position measurement signal coming from SLAM in the *y* direction. Experimental results are illustrated by the residual signal during the flight with detection thresholds, the fault injection and detection times, and a trajectory graphs with *x* and *y* positions. The residual signal used for the fault detection is shown in Figure 6.

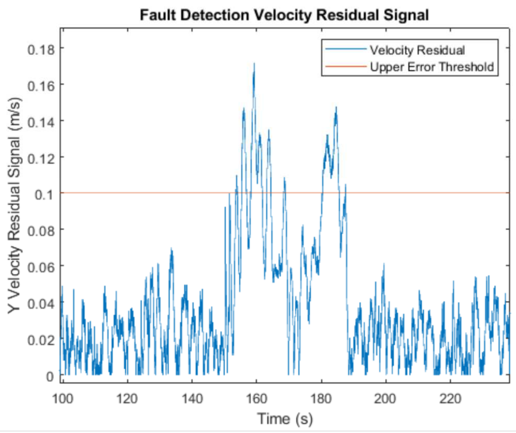

Figure 6 Fault detection residual signal with thresholds

Specifically, during the flight test, a fault in the *y* axis of the SLAM measurement was injected at *t* = 148.3 seconds and removed at *t* = 187.3 seconds. The residual signal used for fault detection is shown in Figure 6. As can be seen, the fault is successfully detected at approximately *t* = 150.2 seconds. After the fault is removed, the residual signal remains within the threshold, allowing detection of the recovery of the SLAM measurement. Figure 7 shows specific timing of fault detection and injection during the flight, including the time instants for fault injection, fault detection, fault removal, and detection of recovered SLAM signal.

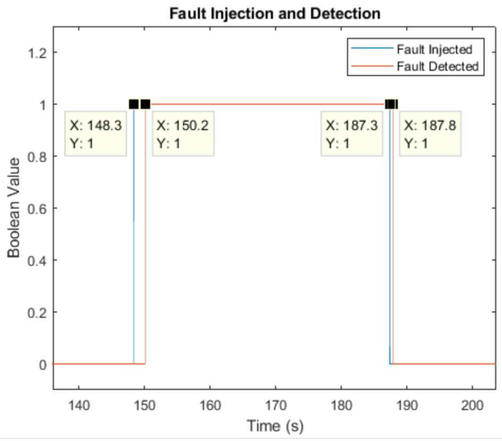

Figure 7 The fault injection and detection signals and significant times

D. Fault-Tolerant Position Tracking

Figure 8 shows the desired flight trajectory and the actual flight trajectory by the UAV before fault detection. Specifically, the position of the UAV in the x and y plane for 98.79 ≤ *t* ≤ 150.2 seconds is shown. Around the 148.3 second mark, as shown in the figure, the fault is injected which results in a small drift between the actual and desired trajectories for 148.3 ≤ *t* ≤ 150.2. Shortly after this drift, the fault is detected at *t* = 150.2 seconds.

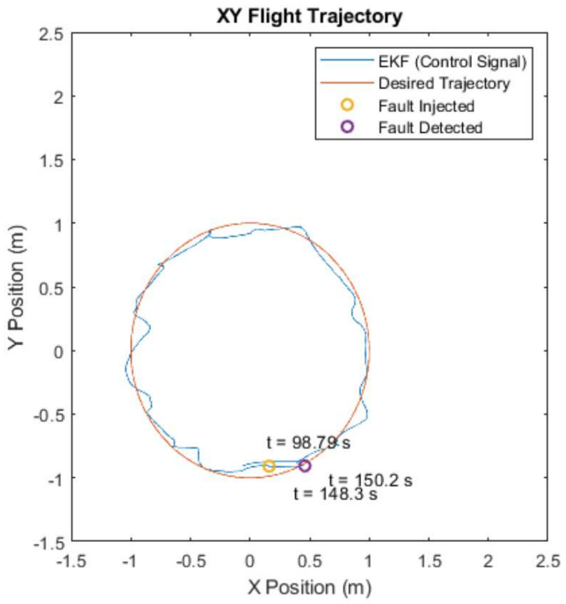

Figure 8 A top-down view of the flight trajectories before fault

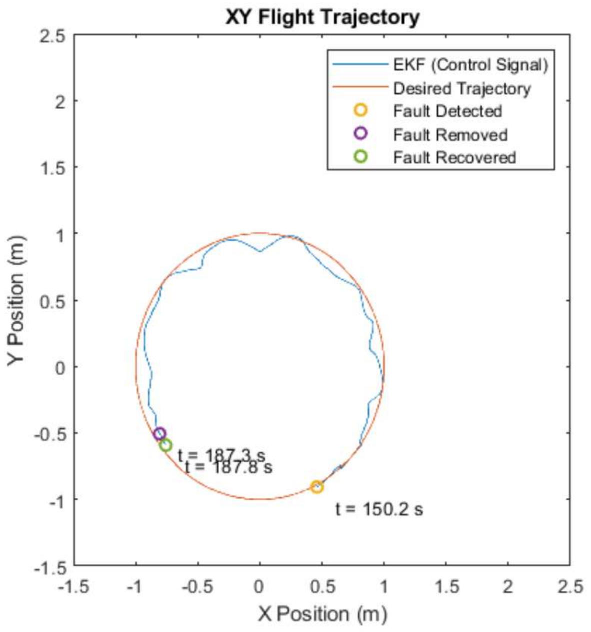

Figure 9 A top-down view of the flight trajectories during fault

Then, the position measurement in the y direction is switched from SLAM to optical flow. Figure 9 shows the flight trajectory utilizing optical flow (i.e. for 150.2 s $\leq t \leq$ 187.8 s). The vehicle is stable, and acceptable tracking performance is maintained. It is detected that the SLAM measurement was recovered at t = 187.8 seconds. As shown by the figure, there is a slight delay between the time when the fault is removed and the time when the recovery of the SLAM signal is detected. Once the healthy SLAM position measurement is detected, it is utilized by the controller again. Figure 10 shows the actual trajectory and desired trajectory for $t \geq$ 187.8 seconds. Nearly a full circle is flown before the landing. The position estimates provided by the fault-tolerant control scheme is shown in Figure 11. For comparison purpose, the true positions (Vicon measurements) are also shown.

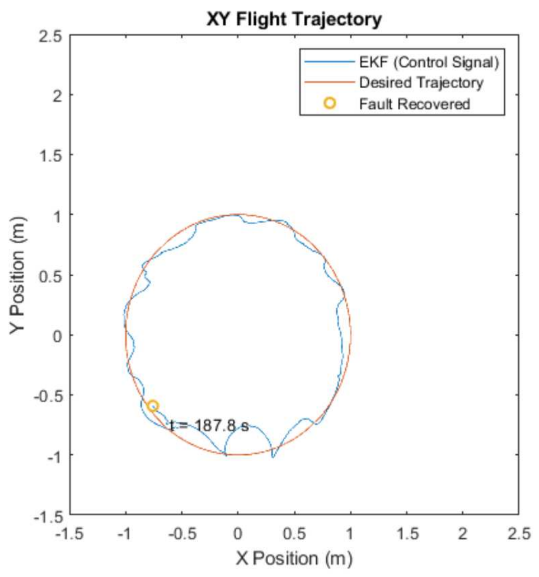

Figure 10 A top-down view of the flight trajectories after fault recovery

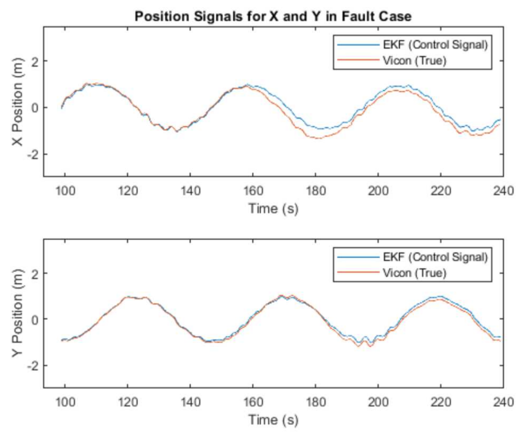

Figure 11 fault-tolerant estimates of x and *y* positions

VI. Conclusion

A fault-tolerant SLAM system for a quadrotor position control using onboard sensors is presented in this paper. Results of SLAM showed pose estimates were very accuracy when compared to the measurements provided by the Vicon motion capture system. A fault detection method is developed to detect faults in the SLAM measurements due to lack of features in the environment (e.g., long featureless hallways or tunnels). Once the fault is detected, the fault-tolerant control method is able to stabilize the quadrotor and maintain acceptable tracking performance using an optical flow localization method. Real-time experimental results have shown effectiveness of the presented fault-tolerant control scheme. Future research may include lowering computational intensity, three-dimensional SLAM, and higher resolution mapping.

VII. Acknowledgment

This work was supported in part by the U.S. National Science Foundation under Grant EEC-1659813.

References

[1] S. Shen, N. Michael, and V. Kumar. "Autonomous multi-floor indoor navigation with a computationally constrained MAV." *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011.

[2] S. Grzonka, G. Grisetti, and W. Burgard. "A fully autonomous indoor quadrotor." *IEEE Transactions on Robotics* 28.1 (2011): 90-100.

[3] W. Aguilar, G. Rodriguez, L. Alvarez, S. Sandoval, F. Quisaguano, A. Limaico, "Visual SLAM with a RGB-D camera on a quadrotor UAV using on-board processing." *International Work-Conference on Artificial Neural Networks.* Springer, Cham, 2017.

[4] Keshavan, Jishnu, et al. "Autonomous vision-based navigation of a quadrotor in corridor-like environments." International Journal of Micro Air Vehicles 7.2 (2015): 111-123.

[5] Avram, Remus C., Xiaodong Zhang, and Jonathan Muse. "Quadrotor sensor fault diagnosis with experimental results." Journal of Intelligent & Robotic Systems 86.1 (2017): 115-137.

[6] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, D. Nardi, "Fast and accurate SLAM with Rao–Blackwellized particle filters." Robotics and Autonomous Systems 55.1 (2007): 30-38.

[7] G. Grisetti, C. Stachniss, and W. Burgard. "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling." Proceedings of the 2005 IEEE international conference on robotics and automation, 2005.

[8] Burton, Andrew, Radford, John. "*Thinking in Perspective: Critical Essays in the Study of Thought Processes.*" Routledge.

[9] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674.

[10] *Quanser QDrone User Manual,* https://www.quanser.com/products/qdrone