Navigation for an Autonomous Helicopter using Onboard Sensors

José Esteves
Department of Electrical and Computer Engineering
Instituto Superior Técnico
1049-001 Lisboa, Portugal
Email: joseesteves@tecnico.ulisboa.pt

Nils Axel Andersen
Department of Electrical Engineering
Danmarks Tekniske Universitet
2800 Kgs. Lyngby, Denmark
Email: naa@elektro.dtu.dk

Pedro Lima
Department of Electrical and Computer Engineering
Instituto Superior Técnico
1049-001 Lisboa, Portugal
Email: pal@isr.tecnico.ulisboa.pt

***Abstract*—The localization and navigation of Unmanned Aerial Vehicles (UAVs) in indoor environments is a research topic of interest, given self-contained approaches still present limitations that require to take into consideration the surroundings, computational capabilities, and energy and weight capacities.**

This dissertation introduces a solution for a UAV to autonomously execute a straight path in a known indoor environment, which was implemented in a six rotor UAV. This implementation provides to an inexperienced user the ability to safely define the path of the hexacopter, without any sensor or computation external to the UAV. Our approach relies on an Inertial Measurement Unit (IMU), an ultrasonic sensor, a barometer and a laser range finder (LRF) to compute robust position and attitude estimates. Simulink control models were developed to guide the parameter tuning of the real controllers.

A fault-tolerant Extended Kalman Filter (EKF) is built to estimate the vertical position and velocity of the hexacopter; the filter is robust to barometer or sonar outages and can detect ground-level changes. A cascaded EKF is used to estimate xy position and velocity, where both the EKFs perform prediction and update steps, but one focuses on computing a 2-D pose estimate using LRF measurements, whereas the other uses these pose estimates to correct the state propagated using the IMU. Although this approach does not guarantee convergence, it proved to be effective in practice. The final system is controlled remotely by sending position and yaw setpoints; other control methods are available. The hexacopter is able to execute a straight path, hover autonomously for an indefinite amount of time in a bounded space and is stable against force or torque disturbances.

I. Introduction

Micro Aerial Vehicles (MAVs) are becoming essential in many important situations. They can be used as a solution for inspection of closed places dangerous for a human being to enter, such as: contaminated houses, factories or power plants, buildings whose structure is at risk of falling down, mining sites with the risk of explosion or rupture. They are also a great candidate for monitoring buildings, delivering packages inside hospitals, factories and workplaces, and, ultimately, for the enjoyment and/or aid inside the house of the everyday-user.

The motivation for this project comes from the need for the deployment of these autonomous vehicles in indoor structured environments, where common localization solutions, such as the Global Positioning System (GPS), are not available and where it is not plausible to plant external sensors or artificial markers on the environment, since these are not scalable options. For the localization of a MAV in a small closed space, not only the accuracy and sampling rate of the sensors should be taken into consideration, but also environment the robot is inserted in. For research purposes, there are several alternatives to compute the localization of robots: Real-Time Location System (RTLS), Computer Vision using external fixed camera(s), Motion Capture System with Passive or Active markers. Some of these approaches are able to give position and velocity estimates accurate and fast enough to control a MAV in an indoor setting. However, these do not represent a scenario that can be expanded in large scale. Approaches relying on on-board computer vision algorithms are widely used, but they are either too computationally expensive to give real-time estimates or simpler methods may not provide absolute pose estimation with bounded error, giving only relative motion estimates. Advances are being made to perform computer vision tasks that can estimate absolute localization, with limited computational power.

Self-localization cannot be solved relying entirely on inertial information or motion measurement because it leads to unbounded position error. In this project, we rely on a LRF, which only recently became lightweight enough to be carried by a MAV, to sense the environment. An ultrasonic sensor, the obvious choice to measure altitude above ground, is used. Furthermore, an IMU is essential for providing high frequency estimates that allow to observe fast dynamics.

This project aims at providing a autonomous path execution solution for a generic multirotor in (partilally) known indoor structured environments, focusing on self-localization. This work is in the domain of a localization-only strategy, in the 6DoF space; we do not aim for any exploratory purpose. Localization is the process of finding the position and orientation of the robot, *i.e.*, its pose. To achieve this, we assume to know a simple 2-D metric model of the environment. Estimating the pose in real-time with a good accuracy is essential to achieve an autonomous robot, especially for a flying robot in an indoor environment.

This project was also motivated by the ambition of expanding DTU's software, Mobotware, to interact and control an aerial vehicle, especially to enable the interaction with the Pixhawk [1]. The Pixhawk runs the PX4 autopilot, which lacks the software to autonomously control a MAV indoors. Mobotware and PX4 have never functioned together before. The capabilities of both will be augmented, in order to create a safe way of controlling a MAV indoors by any person, regardless of his/her knowledge on aerial vehicles.

For this project, the final system should fulfill the goals:

- Use only on-board sensors,
- Computation limited by cheap on-board computer and Pixhawk,
- Ability to perform 6DoF self-localization in a (partially) known structured environment without modifying it,
- Move autonomously from one point to another,
- Stay autonomously at the same pose for an indefinite amount of time, inside a confined circle of radius 15cm bigger than radius of the MAV,
- Withstand disturbances, such as a person pushing or rotating it,
- Be safe for a person to be around.

II. Related Work

Vision from on-board cameras can be fused with IMU information to estimate attitude and/or position [2], [3]. When visible landmarks are not available or do not provide enough information, the self-localization problem using vision can be simplified by modifying the environment, adding visual markers [1], [4].

Besides being used on-board, cameras can also be used as external sensors to estimate position and attitude [5]. To enable tracking of extremely difficult maneuvers, such as triple flips [6], expensive motion capture systems are a commonly adopted solution. This method is very efficient to test algorithms that need accurate measurements and to serve as a ground truth reference to evaluate estimation approaches. Lorenz Meier et al. use a motion capture system with very precise estimates (error ¡1mm) at 250Hz, to report the results of the Pixhawk system [1].

Information from several sensors can be combined to achieve exploratory goals. In 2010, SLAM algorithms were still too computationally demanding, even for powerful desktop computers, thus not being implementable on-board on MAVs [7]. Achtelik et al. present their achievements towards a solution that combines an on-board camera and a LRF sensor to autonomously navigate in unstructured and unknown indoor environments [7]. By that time, Blösch et al. had created the first solution to navigate a MAV through an unknown and unstructured environment, using a single camera as exteroceptive sensor to perform Visual SLAM (VSLAM) [8]. To do so, they resorted to a ground computer to run the SLAM algorithm. Grzonka et al. perform SLAM using a LRF, but also resort to off-board computation [9].

Recently, it has become possible to do SLAM using only on-board computation. Shaojie Shen et al. present a self-contained approach to perform SLAM using a LRF and a camera [10]. Another approach uses a novel inertial-optical flow (IFO) [11] to estimate full attitude and almost drift-free metric distance, along with a camera to perform VSLAM [12].

The problem of position estimation in a known (or partially known) two-dimensional (2-D) polygonal environment has been solved for some time; Cox et al. solve it by matching noisy LRF scans against an available metric map of the environment consisting of polygonal obstacles [13]. The method consists of an iterative least-squares algorithm that finds the congruence between a range scan and the map, provided that the initial displacement is small. Jensfelt discusses three laser-based localization approaches [14]: EKF feature-based localization; Multiple Hypothesis Localization; Monte Carlo Localization. Gutmann and Schlegel also compared several approaches for self-localization using a 2-D laser range finder [15]. The most successful techniques for state estimation are Bayesian filters, such as particle filters or extended and unscented Kalman Filters [16]. Several variations of these filters have been studied and compared [17].

To perform a feature-based localization, it is necessary to extract the features from sensor data. Nguyen et al. compare several line extraction algorithms [18], and the fastest method described, Split-and-Merge, has been modified by by Borges et al., creating the Split-and-Merge Fuzzy (SMF) algorithm [19].

In a feature-based localization algorithm, it is important not only to extract features, but also to represent them. A new algorithm for representation of structured environments with low measurement noise is developed by Harati and Siegwart [20]. Features can be separated as straight-line or, for curve segments, as triangles [21].

An alternative to a feature-based approach is an iconic approach, where raw sensor data is used. Iterative Closest Point (ICP) is widely used, and several variants of this algorithm have already been implemented and compared [22]. The ICP algorithm estimates a transformation between two point clouds, which can be used with LRF measurements to describe their relative displacement and rotation.

III. Hardware Platform

The following hardware was used:

A. Multirotor

The multirotor used is a modification of the hexacopter model MK-Hexa2, by Mikrokopter. The original main board, Fligh Management Unit (FMU), was replaced by the Pixhawk's PX4FMU. The hexacopter does not have a battery; its power is supplied by a moving ground station through a tether, which enables very long flights.

B. On-board Computer

A BeagleBone Black is used as a companion computer for the Pixhawk. It is a single-board computer on the lower price range, which is used to read and process information coming from the LRF sensor, computing the localization in the x and y dimensions and heading estimate. The BeagleBone is also responsible for receiving and interpreting high-level user commands.

C. Laser Range Finder

A LRF is the sensor used to enable the localization on the x and y dimensions. The reasons behind this choice are accuracy in the measurements of the environment ($\leq 1\%$ of measurement); low quantity of data, while containing a lot of meaningful information; high-enough sampling rate (10Hz); high angular resolution (0.36^o); range distance high enough for the environments considered (4m). Having a good ratio of information per size of data enables to use fast algorithms and reduces the need to pre-process information, making it possible to do all the computations on-board. The LRF used is the model URG-04LX.

D. Ultrasonic Sensor

The ultrasonic sensor, sonar, used is I2CXL-MaxSonar-EZ4. It has a 10Hz sampling frequency with 1cm resolution and a narrow beam. A hardware filter was constructed to reduce conducted electrical noise coming from the power supply. Even though its accuracy is lower than a 1-D LRF, it is cheaper and gives better results when the hexacopter is slightly tilted, due to the beam shape of the sensor.

E. Pixhawk

The Pixhawk is an open-hardware computer that runs a Flight Management Unit (FMU) called PX4FMU and also manages several built-in sensors and, optionally, external sensors. It originally includes accelerometer, gyroscope and barometer sensors.

IV. Software Platform

The following software was used:

A. PX4 Autopilot

The PX4 is the open-source autopilot chosen to run on the Pixhawk. It is divided into modules that can be modified individually. The modules that manage the main control loop, running at 250Hz, are a position estimation EKF, an attitude estimation EKF, a position controller and an attitude controller.

The Pixhawk has different behaviors that are defined as flight modes. The one necessary to master in order to reach the goals of this project is the Offboard mode; the remaining are left as additional ways of controlling the MAV. The Offboard mode allows the UAV to be controlled from the BeagleBone, which communicates with the Pixhawk using the MAVLink protocol.

In order for the flight modes to be enabled and working correctly, the position and attitude estimators had to be changed. The first was created from scratch, while the latter was slightly modified. These will be addressed in this report. The attitude and position controllers had to be tuned in order to function properly, although still using the original architecture from the PX4.

B. Mobotware

Mobotware is DTU's real-time control software. It is used in this project and modified to the project's needs. This software consists on a central program called Robot Hardware Daemon (RHD) and other applications running in parallel, where the ones used in this project are: Mobile Robot Control (MRC) and ULMS.

1) Robot Hardware Daemon: (RHD) is a real-time synchronized database that runs at 100Hz and contains all the relevant variables that are shared between programs, such as the pose of the robot. RHD needs a plugin to interact with the Pixhawk, hence an RHD plugin called MavlinkComm is created. MavlinkComm can also interface the variable database to communicate with MRC.

2) Mobile Robot Control: (MRC) is a low-level application, running in soft real-time, that is in charge of tasks related to robot control. It is used in this project to:

- manage the localization algorithm running in the ULMS server,
- as an entry point for the user to enter commands,
- manage the information flow between the entities ULMS$\leftrightarrow$RHD and user$\leftrightarrow$RHD.

MRC is responsible for getting information to and from the ULMS server regarding the robot's pose. The final approach taken to deal with MRC uses a Telnet client to control it.

3) Telnet Client: Enables the user to send commands to the hexacopter to:

- change flight mode,
- give x, y and z position setpoints for the hexacopter to go to,
- give yaw setpoints to change the hexacopter's heading.

The commands are sent from a ground-based computer to the BeagleBone, overriding the Remote Control (RC). The Telnet Client has other roles.

4) ULMS Server: ULMS is the name of the server in charge of communicating with the LRF, processing its information and running the algorithms that use such information. It communicates with MRC and exchanges pose information about the robot. A ULMS plugin, *aulocalize*, is used to compute a localization algorithm.

V. Mathematical Model

The physical inertia model is constructed based on a simplified version of the real hexacopter. In this version, the body is symmetric with respect to any plane that is orthogonal to one of the x, y or z axes and that plane contains the center point, which is the center of mass. This simplification makes the Inertia Tensor a diagonal matrix:

$$\mathcal{I} = \begin{bmatrix} 0.0449 & 0 & 0 \\ 0 & 0.0431 & 0 \\ 0 & 0 & 0.0838 \end{bmatrix}. \tag{1}$$

The hexacopter is modeled in the Body Frame using Newton-Euler equations, which provide a global characterization of the dynamics of a rigid body subject to external forces and torques.

$$\begin{bmatrix}\dot{v}_x\\ \dot{v}_y\\ \dot{v}_z\end{bmatrix} = \begin{bmatrix}\dot{\theta}w-\dot{\psi}v-(c_\psi s_\theta c_\phi + s_\psi s_\phi)g\\ \dot{\psi}u-\dot{\phi}w-(s_\psi s_\theta c_\phi - c_\psi s_\phi)g\\ \dot{\phi}v-\dot{\theta}u-c_\theta c_\phi g\end{bmatrix} + \\ + \begin{bmatrix}-\frac{1}{2m}CA_x\rho u|u|\\ -\frac{1}{2m}CA_y\rho v|v|\\ \frac{1}{m}(-\frac{1}{2}CA_z\rho w|w| + \sum_{i=1}^{6} C_T\rho A(\Omega_i R_{prop})^2)\end{bmatrix} \\ \begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}\dfrac{\dot{\theta}\dot{\psi}(\mathcal{I}_{yy}-\mathcal{I}_{zz}) + l\sum_{i=1}^{6}T_i\sin(\frac{i-1}{6}) + J_r\dot{\theta}\Omega_r}{\mathcal{I}_{xx}}\\ \dfrac{\dot{\phi}\dot{\psi}(\mathcal{I}_{zz}-\mathcal{I}_{xx}) + l\sum_{i=1}^{6}T_i\cos(\frac{i-1}{6}) - J_r\dot{\phi}\Omega_r}{\mathcal{I}_{yy}}\\ \dfrac{\dot{\phi}\dot{\theta}(\mathcal{I}_{xx}-\mathcal{I}_{yy}) + \sum_{i=1}^{6}(-1)^iQ_i + J_r\dot{\Omega}_r)}{\mathcal{I}_{zz}}\end{bmatrix} \tag{2}$$

VI. Methodology

A. Preliminary Work

1) Attitude Controller: A cascaded controller is used, with an inner loop PD controller and a outer loop P controller. To study the controller and simulate its response, a Simulink model of the hexacopter was constructed, using a quadrotor model developed by Brogaard [23] as a baseline. The Inertia model constructed in Section V was used in this Simulink model. A model of the hexacopter system linearized around the hover point is used.

Given the similarity between the dynamics of roll and pitch, the same gains were applied on their controllers, even though they are not the same.

The values that produced the best simulated response were obtained. For the angular velocity control loop, these values are K_P^{vel}=0.2 and K_D^{vel}=1.8; K_P^{pos}=6 is selected for the position P gain. Our final controller does not have a large bandwidth (6rad/s), but has a large stability margin, with a 66.5 degree Phase Margin and 11.6dB Gain Margin. The response has a 10%-90% rising time of 181ms.

Experimentally, the values chosen are K_P^{pos}=6.5 , K_P^{vel}=0.11 , K_D^{vel} =0.01 and K_I^{vel}=0. A lower D velocity gain is chosen compared to the simulation value not to sacrifice the bandwidth of the controller.

2) Communication Software: In order to enable the onboard computer (BeagleBone) to be part of the hexacopter's control loop system, the RHD plugin MavlinkComm was designed with the set of features:

- establish a MAVLINK connection [1],
- receive information regarding the hexacopter's attitude and local position,
- send position and attitude error/absolute estimates,
- send command to enable/disable Pixhawk's flying modes,
- send attitude and position setpoints.

3) Algorithm Simulators: Two simulators were created in Matlab to study the implementations of the EKFs presented in Sections VI-B and VI-C for altitude and xy estimation, respectively.

The noise specifications of the sensors in simulation were unfavorably aggravated to assure that the simulator does not present better conditions than reality. The main differences between the measurement models designed and the real sensor measurements are:

- Sonar: fixed attitude is considered, with zero pitch and roll angles. Obstacle detection noise is discarded. The frequency of spikes in the measurements is unfavorably increased comparing to reality.
- Barometer: temporary short-term effects arising from changes on the environment, such as opening a window, are neglected.
- Accelerometer: scaling factor inherent to the sensor is ignored.
- Laser-based estimates: number of line features observed by the laser is assumed to be enough to compute a pose estimate. Fixed attitude is considered, but noise is considerably increased to compensate for this benefit.

4) Attitude estimation: In indoor environments, magnetic fields change abruptly, especially due to materials in the building's structure. This makes the magnetometer an unreliable source of yaw angle measurements. The attitude estimator running on the Pixhawk was modified to use yaw estimates provided by the laser-based localization algorithm implemented, instead of magnetometer measurements. PX4's original estimation of roll and pitch is used.

5) Map setup: The map used as a testbed in all the localization experiments is a corner with two walls with 3m width and height close to 2.2m. These walls do not touch the ceiling, hence it is common to loose track of the walls for 2 or 3 LRF measurements if the hexacopter is very tilted at some point. The performance of the localization algorithm improves if more walls are added to the map. This testbed is chosen to test how the system behaves when using the minimum features required for the localization algorithm to work, which is the the worst case scenario.

B. Study 1: Implementation of the Altitude Estimation

The altitude estimation is performed using an EKF, where its implementation aims at lowering computational complexity, while maintaining good results. Pre-processing information from the ultrasonic sensor is proposed in order to get a signal containing only white Gaussian noise; the barometer signal is filtered to reduce its noise.

1) Sensor Model: The accelerometer measurements, sampled at f_1=250Hz, are modelled as

$$u = a - \alpha a - b_u - \eta_1 \tag{3}$$

where a is the true acceleration and η_1 is a Gaussian white noise process. The scale factor of the accelerometer is neglected, *i.e.*, α=0. The measurement u contains a bias b_u modeled as a constant plus random walk process with uncorrelated white Gaussian noise ω_1:

$$\dot{b}_u = \omega_1. \tag{4}$$

The barometer measurements, at f_2=100Hz, are translated into an altitude above ground measurement and modeled as

$$y_{bk} = p_k + b_y(k) + \eta_2(k) \tag{5}$$

where p_k is the true altitude position at instant k and η_2 represents white Gaussian noise that can be split into two types of Gaussians [24]: a first-order Gauss-Markov random process and uncorrelated random process. The bias $b_y(m)$ is modeled as a slow time-varying signal

$$\dot{b}_y = -\lambda_y b_y + \omega_2 \tag{6}$$

where λ_y is an inverse of correlation time of Gauss-Markov random process for the bias [25], estimated as λ_y =1/100, and ω_2 is uncorrelated white Gaussian noise.

The ultrasonic sensor provides position information sampled at f_3=10Hz and is modeled as

$$y_{sm} = p_m \cos(roll)\cos(pitch) + \eta_3(m) \tag{7}$$

where p_m is the true altitude position at instant m. The discrete-time noise samples $\eta_3(m)$ are white Gaussian noise after the measurement goes through a validation gate.

Having good error models of the sensors allowed us to create certain assumptions in the design of the localization algorithms. We assume that the process and measurement noise covariance matrices converge to steady-state values. If the sensors do not follow the error model that correspond to the KF design, the measurements are rejected. Using these assumptions allows to decrease the complexity of the EKFs greatly, consequently speeding up the computations, since the computational load of aided navigation systems is typically dominated by the time update of the covariance matrix [26].

2) Implementation: Our kinematic model has state $\boldsymbol{x} = \left[p, v, b_u, b_y\right]^T$. Admitting constant acceleration between integration steps of length $\tau = \frac{1}{f_1}$, one gets the Transition State Model, sampled at $t_k = k\tau$, and Control Input Matrix:

$$\boldsymbol{\Phi}(\tau_i, \tau_{i-1}) = \begin{bmatrix} 1 & \tau & \frac{1}{2}\tau^2 & 0 \\ 0 & 1 & \tau & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-\lambda_y \tau} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \frac{1}{2}\tau^2 \\ \tau \\ 0 \end{bmatrix}, \tag{8}$$

where our input is the accelerometer measurement in the z direction. The state is propagated at f_1=250Hz using

$$\hat{\boldsymbol{x}}_{\tau_i} = \boldsymbol{\Phi}(\tau_i, \tau_{i-1})\boldsymbol{x}_{\tau_{i-1}}. \tag{9}$$

The observation matrix can be written as $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ for the barometer and sonar, respectively, as presented in (10).

$$\boldsymbol{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{H}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

The sonar measurement of velocity is the derivative of consecutive valid sonar measurements. The correction of b_y is computed when a valid sonar measurement is received, with

$$z_{b_y} = (y_{bk} - \hat{b}_y^-) - y_{sm}, \tag{11}$$

where y_{bk} is the latest barometer measurement, $\hat{b}_y^-$ is the barometer bias estimate prior to update and y_{sm} is the new sonar measurement. Only the slow time-varying part of the divergence of measurements between sonar and barometer is used to correct b_y. The sonar is the main responsible for correcting b_y, hence the Kalman Gain associated with this correction is treated as part of the sonar's Kalman Gain vector.

The measurement correction is computed with (13) when there is a new sonar or barometer measurement available, but this correction is spread out along the iterations where no measurement is available. The correction is applied in every iteration, at 250Hz, using (12).

$$\hat{\boldsymbol{x}}_n^+ = \hat{\boldsymbol{x}}_n^- + \boldsymbol{K}\boldsymbol{z}_n\tau, \tag{12}$$

$$\text{with } \boldsymbol{z}_n = \boldsymbol{y}_n - \hat{\boldsymbol{y}}_n^-, \tag{13}$$

where $\boldsymbol{y}_n^-$ can be either $\boldsymbol{y}_k^- = \boldsymbol{H}_1\hat{\boldsymbol{x}}_k^-$ in case the position is corrected with a barometer measurement or $\boldsymbol{y}_m^- = \boldsymbol{H}_2\hat{\boldsymbol{x}}_m^-$ if the state vector is corrected with sonar information.

The barometer and sonar Kalman Gains, $\boldsymbol{K}_b$ and $\boldsymbol{K}_s$, were manually tuned taking into consideration the measurement noise and experimental results.

$$\boldsymbol{K}_b = \left[0.1, 0, 0, 0\right]^T \qquad \boldsymbol{K}_s = \left[4, 1.5, 4, 0.5\right]^T \tag{14}$$

The barometer correction is only applied on the estimator as a fault tolerant approach, if the sonar is not valid for more than 0.3s. The barometer corrections will be deactivated again when a new valid sonar measurement is received.

C. Study 2: Implementation of the XY Estimation

A cascaded Extended Kalman Filter (CEKF) is used to estimate the position and velocity of the hexacopter in x and y dimensions. To reach this goal, other state variables have to be estimated, such as the biases of the accelerometer sensor. The CEKF approach combines one EKF running on the Pixhawk, Pixhawk's EKF (PEKF), and another running on the Beaglebone, BeagleBone's EKF (BEKF). The first, PEKF, is similar to the altitude EKF presented in Section VI-B, where the state propagation is done at 250Hz, using IMU information, and the update step is computed at a lower frequency of 9Hz. The second, BEKF, uses a line matching algorithm to compute the robot's pose based on a 2-D map and LRF measurements; it provides position estimates to the PEKF at 9Hz. The PEKF provides dead-reckoning navigation for a short period if the BEKF is not running.

1) Sensor Model: The sensors used in the CEKF are an accelerometer, modeled as (3), and a LRF. The LRF measurements are not modeled, since they are not used as a KF input, only features are extracted from LRF measurements; instead the output of the BEKF, sampled at f_2=9Hz with a 0.22s delay, is modeled as

$$y_k = p_{k-2} - \eta(k) \tag{15}$$

where p_{k-2} is the true position delayed by 0.22s and $\eta(k)$ is a Gaussian white noise process.

2) Implementation:

a) PEKF: The EKF running on the Pixhawk, PEKF, is responsible for estimating all the state variables related to the x and y dimensions, which are fed to the position controller.

In our sensor-based model, the kinematic system has state $\boldsymbol{x} = \left[p_x, v_x, b_{ux}, p_y, v_y, b_{uy}\right]^T$. The Transition State Model, sampled at $t_k = k\tau$, and Control Input Matrices, for each accelerometer, are:

$$\boldsymbol{\Phi}(\tau_i, \tau_{i-1}) = \begin{bmatrix} 1 & \tau & \frac{1}{2}\tau^2 & 0 & 0 & 0 \\ 0 & 1 & \tau & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \tau & \frac{1}{2}\tau^2 \\ 0 & 0 & 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$

$$\boldsymbol{B}_1 = \left[\tfrac{1}{2}\tau^2, \tau, 0, 0, 0\right]^T \quad \boldsymbol{B}_2 = \left[0, 0, 0, \tfrac{1}{2}\tau^2, \tau, 0\right]^T, \quad (17)$$

where our input in $\mathbf{B}_1$ and $\mathbf{B}_2$ is considered to be accelerometer measurements in axes x (u_x) and y (u_y), respectively. The state is propagated, during the prediction step, by (9) at f_1=250Hz.

In our cascaded approach, the output of the BEKF is used as a measurement input in the PEKF, using the observation matrix written in (18). The measurement correction is computed as shown in (19).

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (18)$$

$$\boldsymbol{z}_n = \begin{bmatrix} y_{xk} & \dot{y}_{xk} & y_{xk} & y_{yk} & \dot{y}_{yk} & y_{yk} \end{bmatrix}^T - \mathbf{H}\,\hat{\boldsymbol{x}}_k^- \quad (19)$$

The approach of correcting the state continuously at a high frequency using low-frequency information is used, with (12) applied at 250Hz and (13) applied at 9Hz. This eliminates small discontinuities in the state variables that originated in state updates. The Kalman Gain vectors used in this approach are presented in (20).

$$\begin{aligned} \boldsymbol{K}_x &= \left[2.8, 0.49, 16, 0, 0, 0\right]^T \\ \boldsymbol{K}_y &= \left[0, 0, 0, 2.8, 0.49, 16\right]^T \end{aligned} \quad (20)$$

The implementation had a Kalman Gain associated to the accelerometer biases 4 times higher than wanted. This made the acceleration bias estimation very susceptible to the accuracy of the laser-based position estimates received from the BEKF. The result of the over-correction of the bias can often be detected in the position estimation as an oscillation after the robot moves.

The output position estimates of the BEKF arrive to the PEKF with a delay varying from 0.1s to 0.25s. With this delay, there is not a need to have a special approach to estimate position. In order to get noise-free velocity estimates from delayed BEKF position estimates, the filtering necessary resulted in delays close to 0.5s. It is desired a velocity estimation approach that gives a response as fast as the one from the accelerometer, hence a technique [27] was used to fuse delayed measurements in the EKF. The present state, at t_{now}, is updated by applying a correction that is computed based on the past state error, at $t_{now} - \Delta t_{delay}$, which is the time instant from when the delayed velocity measurement is originally from.

Scaling effects arise from the filtering performed while computing velocity estimates from BEKF position estimates. To prevent its effect on the state update when the robot has fast motions, the velocity correction is saturated.

Velocity estimates are the most important for the stability of our control system, since they will be responsible for the control of the faster responses of the system. Hence the velocity state has to be accurate in real-time, with the minimum delay possible. Position estimates are obviously important, but their accuracy is not as important for the stability of the system. The most important concern regarding the PEKF's position estimates is for them to be accurate enough when providing them to the BEKF as input, so that this filter is able to converge to the real position of the robot. If the PEKF's position estimates are too far from the true value (rule of thumb would be more than 0.3m), the BEKF may not be able to use the LRF information to compute an accurate position estimate (on that iteration) or even converge to the correct position estimate.

b) BEKF: The IMU information is very important to propagate the system state, since it provides very fast estimates and can detect high-frequency dynamics; moreover dead reckoning is simple and inexpensive. Although, the uncertainty of the robot's pose increases with time and, especially, as it moves. In order to reduce the tracking error on the PEKF, it is used a feature-based self-localization algorithm [28], where the features are line segments. The algorithm handles the LRF measurements in order to predict the current absolute pose of the hexacopter in a 2-D space. To initialize this algorithm, the initial pose of the robot in the EF is needed, together with the corresponding variance. In each iteration, the algorithm uses as inputs: pose estimates (optional), LRF measurements and a 2-D metric map of the environment $\mathcal{M}$, described as $\mathcal{M}$=$\{L_1$,L_2,...,$L_m\}$, where L_j represents the line segment with origo O_j, angle ψ_j and length l_j in the EF.

This algorithm is computed in the BeagleBone, hence being named as BeagleBone's EKF, BEKF. It is taken advantage on the fact that man-made indoor environments are generally structured, commonly containing straight walls and fixed obstacles with straight surfaces, which can be well represented by line segments; obstacles that cannot be well represented as such are neglected, and the measurements of these obstacles, collected by the LRF, will be rejected as outliers in our model.

The algorithm is divided in 5 sequential steps:

1) Pose prediction: compute MAV's predicted pose $\hat{\boldsymbol{p}}_i^-$;
2) Observation: extract features from range measurements;
3) Measurement Prediction: Predict observed features based on $\mathcal{M}$ and predicted pose;
4) Matching: Match observed features with corresponding predicted features and compute associated innovation;

5) Estimation: Compute the Update step to estimate the current pose of the robot, $\hat{\boldsymbol{p}}_i^+$. The output can also be a displacement $\Delta\hat{\boldsymbol{p}}_i^+$ representing the predicted pose error.

This algorithm is feature-based, where line segments are extracted from sensor data using RANSAC. Among many geometric primitives, line segment is the simplest one and accurately describes most office environments [18]. RANSAC is evaluated as having bad speed and correctness, but produces more precise lines than other line extraction algorithms and its accuracy can be increased by performing more iterations [18]. By using RANSAC, outlier rejection of small curved features is done spontaneously, thus people detected by the LRF will automatically be rejected as outliers.

One handicap of the algorithm is that, on Step 3, it is not able to distinguish between the features that the robot can physically observe with the LRF, from all the features that are close enough (in $\mathcal{M}$) to be considered inside the range of sight. Depending on the map, this leads to wrong predicted features, which may result in problems in the matching step, if wrongly predicted features are matched to observed features.

There should be features along more than one axis on the environment map. In order for the localization algorithm to work, it is required at least two line features that create a certain angle between them, ideally bigger than 20^o, in any circle of 4m radius in the area where the localization algorithm is computed.

VII. Results

In the following experiments, a source for ground truth is not available. For altitude position, the reader can rely on the sonar position measurements to have a notion of reality, but for altitude velocity the sonar estimates are not as reliable, since they correspond to the differentiation of the sensor's position measurements.

On the xy localization results, it will only be shown results from the y dimension, since x and y variables are estimated exactly in the same way and exhibit the same behavior.

Due to the lack of ground truth, simulations of the algorithms are executed in order to perform error analysis. The simulation scenario is unfavorably worse than reality.

A. Original Solution

1) Altitude Original Solution: An altitude estimation algorithm is available on the PX4 Autopilot. Figures 1 and 2 show its altitude's position and velocity estimation. It is not appropriate for indoor environments, since it only uses barometer and accelerometer measurements. It is targeted for outdoor use, where the barometer can be combined with GPS to provide altitude estimates. In indoor environments, GPS is not available and the barometer by itself does not provide results with enough accuracy or resolution. By comparing to the sonar measurements, one can see that the algorithm does not accurately provide absolute height measurements and its error grows with time due to the barometer's varying bias.

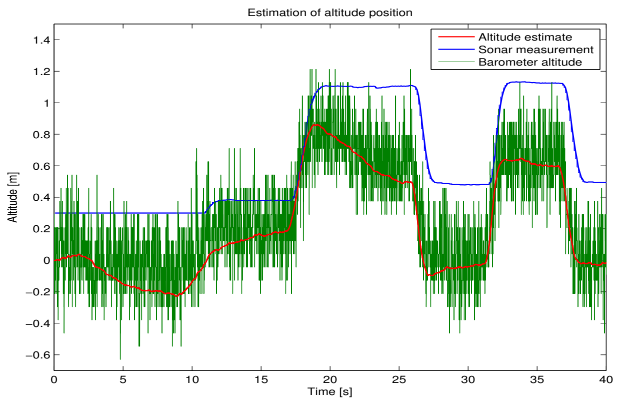

Fig. 1. Estimation of z-dimension position using original PX4's EKF.

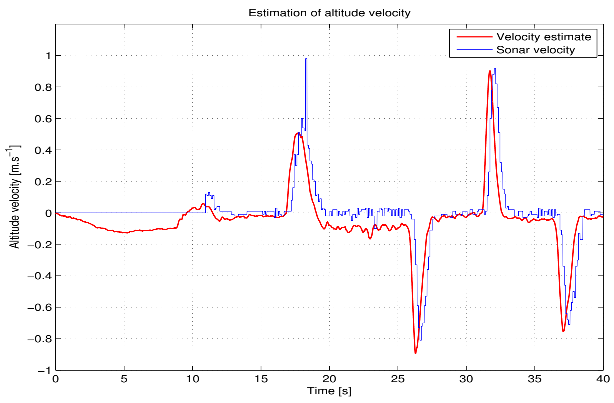

Fig. 2. Estimation of z-dimension velocity using original PX4's EKF.

2) xy Original Solution: The result of the original Pixhawk's localization algorithm is shown in Figures 3 and 4. A PX4FLOW module is added to the MAV in order to perform this experiment. The Optical Flow module provides velocity measurements in the x and y dimensions, based on tracking features observed by the camera of the module. This measurements are used to correct the velocity state and the accelerometer's bias state. In this experiment, the MAV is handheld and moved back and forth between positions x=0m and x=1m. The MAV is left motionless at each point for at least 5s, but drift in the position estimate can be observed. The position estimate drifts 10cm in the interval $t \in$ [17,22]s and in $t \in$ [24,28]s, in Figure 3. The problem is that the bias of the accelerometer is not estimated well-enough using only this sensor, so the state is propagated incorrectly.

A study of Optical Flow sensors used for UAV navigation is done by Gageik et al. [29], from where we can witness several problems in position and velocity estimation, such as bias in the position estimate and scaling issues. Honegger et al. present the need for illumination when using Optical Flow sensor, and the PX4FLOW sensor illumination requirements are described [30]. It is always assumed that no moving objects appear in the camera's field of view, but that is an issue to be taken into consideration, since it may lead to measurement error. The most obvious problem related to the goal of this project is that it is not possible to extract absolute position measurements from Optical Flow measurements.

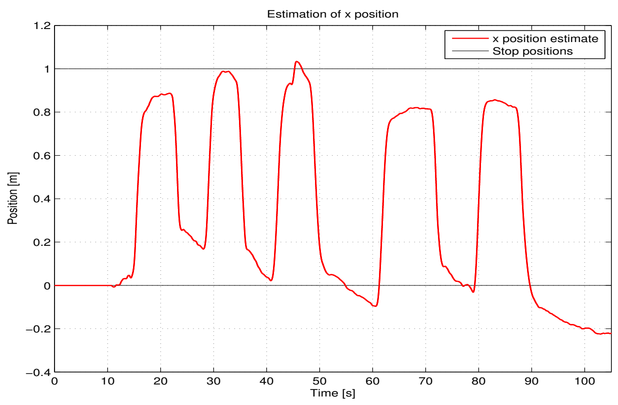

Fig. 3. Estimation of x-dimension position using original PX4's EKF. The experiment is performed holding manually the MAV from the starting point and a point 1m away multiple times. The MAV is left motionless at each point for at least 5s, but drift in the position estimate can be observed.

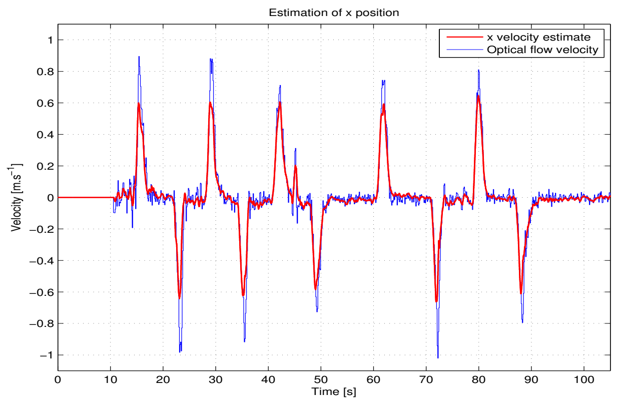

Fig. 4. Estimation of x-dimension velocity using original PX4's EKF.

B. Proposed Solution

1) Altitude Solution: Experiments demonstrate that the altitude estimator can withstand short periods (at least 4s) of sonar outages with bounded error. Figures VII-B1 and VII-B1 show the estimation result during flight using the final EKF. One can see that the estimates predict the sonar measurements. Both the position and velocity estimates are better than the measurements taken from the sonar. Sonar spikes or inaccurate measurements have little or no effect at all on the estimation. Changes of ground level are detected.

An experiment is performed in the simulator where the robot follows a sinusoidal altitude reference with 0.4m amplitude. Figure 7 shows the plot of the position estimate's error along the experiment in the simulator. The error is very low, with a Standard Deviation (SD) of $\sigma = 0.012241$m and maximum error below 0.03m, excluding the initial phase where the algorithm is converging to the initial state ($t \in [0, 0.5]$s). The coefficient of determination R^2 of the altitude estimate in relation to the real estimate is 0.9957, which indicates a very good correspondence between the observed and true positions. The estimates are delayed, which creates a bias on the residuals that changes with state variations. This bias is the sinusoidal component that can be easily spotted on Figure 7. The residuals of the EKF estimation are white Gaussian noise, with ditribution $\mathcal{N} \sim (0, 0.0028^2)$, if the delay is removed (by shifting the estimates 0.068s), which means that the state is observed correctly, only delayed by 0.068s.

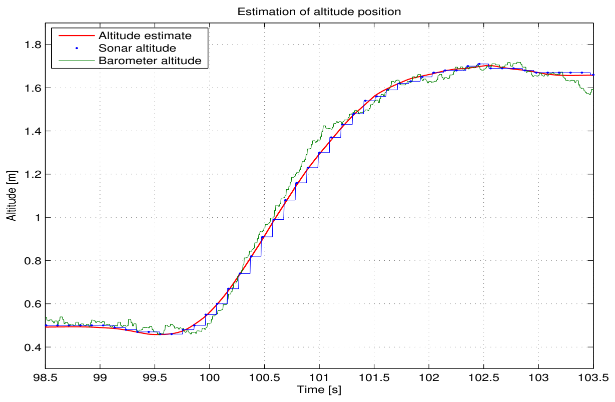

Fig. 5. Altitude position estimation using implemented EKF when the hexacopter is flying, controlled with a RC.

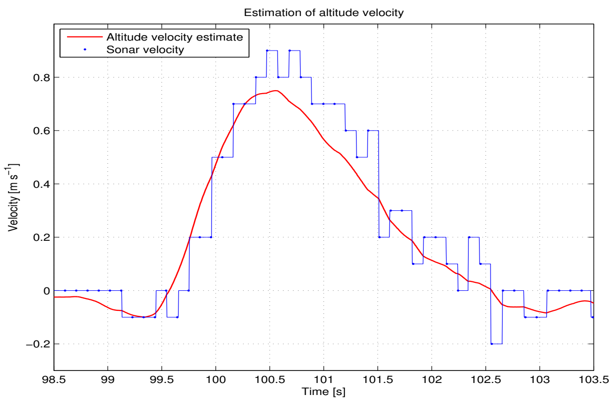

Fig. 6. Altitude velocity estimation using implemented EKF when the hexacopter is flying, controlled with a RC.

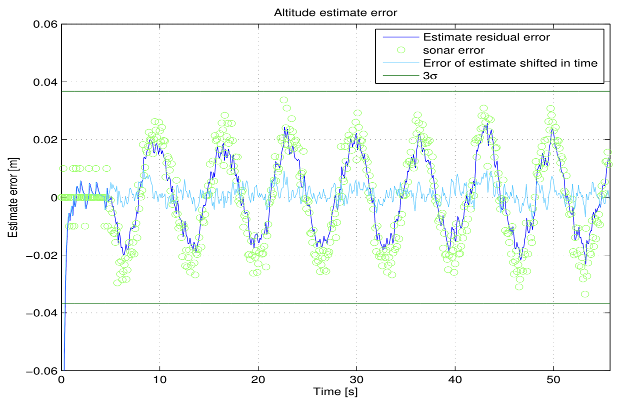

Fig. 7. Error analysis of altitude position estimation using the altitude EKF.

Figure 8 presents the velocity estimate's error (maximum error below 0.05m.s^{-1} and $\sigma = 0.019258\text{m.s}^{-1}$), which is considerably lower than of the estimates received from the sonar (maximum error can go up to 70m.s^{-1} and $\sigma = 0.067175\text{m.s}^{-1}$). The sinusoidal pattern observed ($3-4\%$ of the true velocity) on the estimate's error does not disappear by shifting the signal in time. This varying bias is explained by a small scaling factor between velocity estimate and true velocity; when the velocity is high, the estimation error grows. The R^2 coefficient of fitting the altitude velocity estimate to the ground truth is 0.99301.

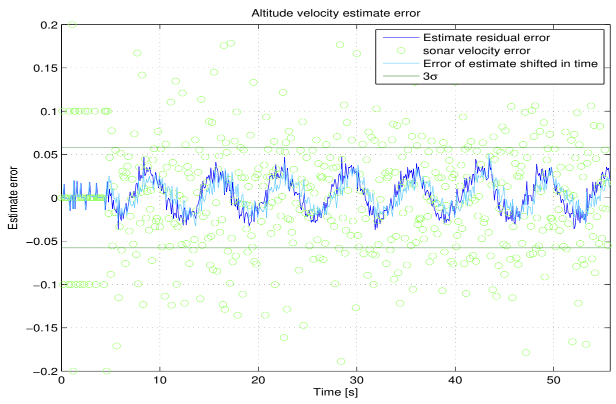

Fig. 8. Error analysis of altitude velocity estimation using the altitude EKF.

2) xy Solution: Even though the BEKF position estimates have a small delay, a good estimation of position is accomplished. The state propagation has almost zero delay. The velocity estimates computed from position estimates received from the BEKF give innacurate measurements in terms of scale, as seen in Figure10 at $t = 82.5$s; filtering the signal is the main cause. The Kalman Gain associated to the accelerometer bias was set bigger than desired. Sometimes the effect of over-correcting the bias can be noticed, as happens after $t = 83.5$s in Figure 9: a small oscillation, and in Figure 10: a positive overshoot.

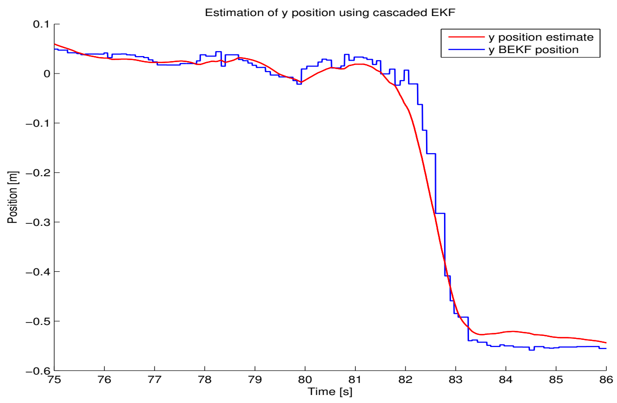

Fig. 9. Estimation of position in y dimension using cascaded EKF, holding the hexacopter manually.

An error analysis is performed with the results collected from the experiment in the simulator where the robot follows a sinusoidal position reference along the x axis. Figure 11 shows the residuals of the position estimates. The error is never bigger than 0.05m and has a SD of $\sigma = 0.020597$. The sinusoidal pattern that the residuals exhibit is mitigated if the signal is shifted 0.14s, where the error becomes approximately normally distributed with variance $\sigma^2 = 0.088^2$. This shows

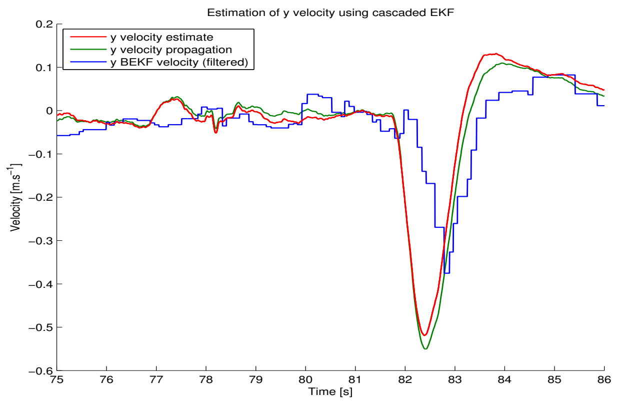

Fig. 10. Estimation of velocity in y dimension using cascaded EKF, holding the hexacopter manually. The effect of over-correcting the accelerometer bias can be noticed after $t = 83.5$s.

that the estimate is correctly estimated, but is delayed 0.14s, which originates on the laser-based estimates having a delay of 0.22s. The estimation delay in simulation is greatly bigger than the observed in reality.

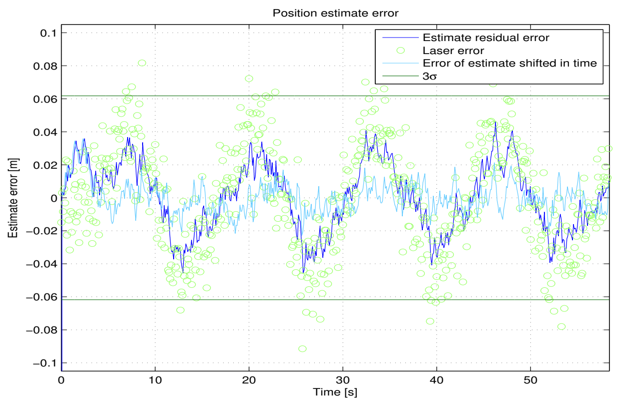

Fig. 11. Error analysis of x position estimation using the CEKF.

The error associated to the velocity estimation is presented in Figure 12. It is smaller than 0.04m.s^{-1}, with a SD of $\sigma = 0.018586$. For the velocity estimate case, shifting it on time will not lead to a normally distributed error, because there is a scaling factor between the true velocity and the estimated one. Our aproach for correcting the velocity only uses laser-based velocity estimates from the past, *i.e.*, the velocity is corrected with a certain delay. This leads to small scaling errors that are created by propagating the state with the acceleration integration. Nevertheless, the error is very small in value, as intended, even with the noise model of the laser-based estimates increased in simulation. The laser velocity estimates have a residual with $\sigma = 0.063274$. The overall fit of the velocity estimates to the true velocity curve has coefficient of determination R^2 $= 0.97054$.

The experimental accuracy of the algorithm cannot be verified due to the lack of ground truth. Unlike the altitude case, we do not have a sensor of which we are certain of its accuracy and that measures the position (or velocity) of the robot directly. From the error analysis of the algorithm, we can conclude that the is estimator is able to provide very

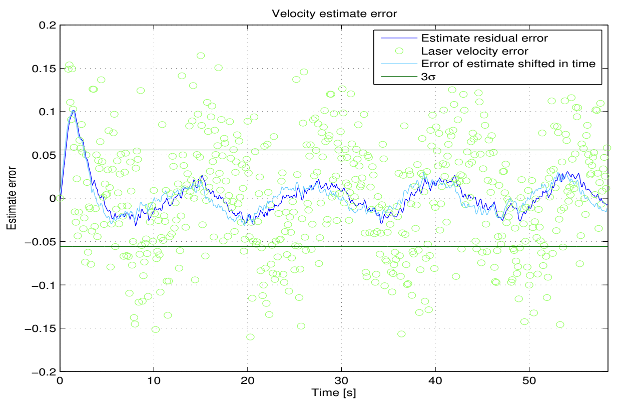

Fig. 12. Error analysis of x velocity estimation using the CEKF. Unlike in the position case, delay on the estimates is not the explanation for the sinusoidal behaviour of the residuals. Instead, it is a scaling factor that is creating errors that depend on the velocity value.

good estimates with noisy measurements, whose noise follows a gaussian distribution. In reality, the accuracy of the state estimation depends on the accuracy of the BEKF position estimates. Hence, the validity of the error analysis performed lies on the accuracy of the model of the BEKF estimates used on the simulator. In motionless scenarios, we can verify that the BEKF provides accurate position estimates, using a measuring tape.

The delay of the estimates is measured against the state propagation curve, which is used as a reference of zero delay. The delay of the estimate is close to zero both in position and velocity. For position, the delay is bigger because the estimates are corrected with a bigger weight by the BEKF estimates; on the velocity estimates, the delay is almost zero, since they are mostly the state propagation itself.

VIII. Conclusion

The work developed and the successful integration of Mobotware and PX4 autopilot, lead to a self-contained control system that is able to autonomously execute a straight line path defined by any user.

Both estimation algorithms are designed to be lightweight and expandable, and achieved very good results. They execute outlier rejection to keep the design assumptions true. The altitude localization algorithm is fault-tolerant, as proposed, while the localization in the x and y dimensions is designed to be easily expandable into a fault-tolerant approach. The xy CEKF localization algorithm ignores people walking around the robot and can withstand LRF measurements that only contain noise in at least 2 consecutive iterations without diverging to incoherent results. The estimation algorithms perform online self-calibration of the barometer and accelerometer sensors. The altitude estimation EKF is able to detect ground level changes bigger than 20cm. The xy localization algorithm has requirements that have to be met in order to achieve successful localization; these are stated in Section VI-C2. The minimum requisites are used experimentally and enable good results. The laser-based localization is computed in a single-board computer in the lower market price range.

The hexacopter is able to stay hovering in a confined circular space with radius 15cm larger than its body. The system is able to be dragged and rotated and it will return by itself to the pose previously ordered by the operator.

There are many topics to be improved. Furthermore, this project can be viewed as a self-localizing autonomous platform that can be used in a future application or for future research.

One improvement to the localization in the x and y dimensions should be emphasized: include in the PEKF the error covariance associated to the pose estimates computed in the BEKF. These covariance values should be used in the PEKF, ideally to compute Kalman Gains, but other procedures may be implemented, such as measurement rejection.

The robustness of the estimation can be increased by adding other algorithms that can provide estimates to the PEKF, which would work in parallel with the BEKF. The algorithms that can be added may use the same sensor information, such as computing motion estimates from consecutive LRF readings [31], [7]. To create a fault-tolerant approach, algorithms can be based on new additional sensors, such as the Optical Flow sensor, PX4FLOW [30], or camera for feature-based motion estimation [32], [3]. SLAM algorithms are not recommended for this system, unless the on-board computer is upgraded. It should be considered algorithms that can complement the localization algorithm already implemented, ideally compensating for its limitations; this would increase the safety of the system. The position will only be propagated for 1 second if the BeagleBone stops sending laser-based position estimates to the Pixhawk. After that, the system was designed so that the Pixhawk denies being able to estimate x and y position and velocity, consequently entering fail-safe mode.

Now that the system is able to localize and execute a defined path autonomously, the navigation solution can be improved to avoid obstacles, in order to increase safeness. Trajectory planning can be added, leaving the role of the user even more futile. Implementations of path planning with obstacle avoidance are already available [33], [9]. Other approaches can be followed using our existing case, where a starting map is given by the user, which is then modified and expanded online to match the real environment more accurately.

The system developed can also be used as a testing platform. Due to the cascaded approach adopted for the localization in the x and y dimensions, the BEKF localization algorithm can easily be substituted by another approach. This allows to fulfill a performance study of several approaches.

References

[1] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “Pixhawk: A system for autonomous flight using onboard computer vision,” in *ICRA'11*, 2011, pp. 2992–2997.

[2] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, “Monocular vision for long-term micro aerial vehicle state estimation: A compendium,” *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013. [Online]. Available: http://dx.doi.org/10.1002/rob.21466

[3] K. Schauwecker, N. Ke, S. Scherer, and A. Zell, “Markerless visual control of a quad-rotor micro aerial vehicle by means of on-board stereo processing,” in *Autonomous Mobile Systems 2012*, ser. Informatik aktuell, P. Levi, O. Zweigle, K. Huermann, and B. Eckstein, Eds.

Springer Berlin Heidelberg, 2012, pp. 11–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32217-4_2

[4] G. P. Tournier, M. Valenti, J. P. How, and E. Feron, "Estimation and control of a quadrotor vehicle using monocular vision and moir patterns," in *In AIAA Guidance, Navigation and Control Conference*. AIAA, 2006, pp. 2006–6711.

[5] E. Altug, J. Ostrowski, and C. Taylor, "Quadrotor control using dual camera visual feedback," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 3, Sept 2003, pp. 4294–4299 vol.3.

[6] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 1642–1648.

[7] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments," in *Proceedings of SPIE*. SPIE, 2010, p. 7332. [Online]. Available: http://dspace.mit.edu/handle/1721.1/52660

[8] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 21–28.

[9] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 90–100, Feb 2012.

[10] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained mav," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 20–25.

[11] S. Weiss, R. Brockers, and L. Matthies, "4dof drift free navigation using inertial cues and optical flow," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 4180–4186.

[12] R. Brockers, M. Hummenberger, S. Weiss, and L. Matthies, "Towards autonomous navigation of miniature uav," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, June 2014, pp. 645–651.

[13] I. Cox and J. Kruskal, "On the congruence of noisy images to line segment models," in *Computer Vision., Second International Conference on*, Dec 1988, pp. 252–258.

[14] P. Jensfelt, "Approaches to mobile robot localization in indoor environments," PhD thesis, Signal, Sensors and Systems (S3), Royal Institute of Technology, SE-100 44, Tech. Rep., 2001.

[15] J.-S. Gutmann and C. Schlegel, "Amos: comparison of scan matching approaches for self-localization in indoor environments," in *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on*, Oct 1996, pp. 61–67.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[17] D. C. K. Yuen and B. A. Macdonald, "A comparison between extended kalman filtering and sequential monte carlo techniques for simultaneous localisation and map-building," in *In Proceedings of the 2002 Australasian Conference on Robotics and Automation*, 2002.

[18] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug 2005, pp. 1929–1934.

[19] G. Borges and M.-J. Aldon, "Line extraction in 2d range images for mobile robotics," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 3, pp. 267–297, 2004. [Online]. Available: http://dx.doi.org/10.1023/B%3AJINT.0000038945.55712.65

[20] A. Harati and R. Siegwart, "A new approach to segmentation of 2d range scans into linear regions," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, Oct 2007, pp. 2083–2088.

[21] R. Vázquez-Martín, P. Núñez, A. Bandera, and F. Sandoval, "Curvature-based environment description for robot navigation using laser range sensors," *Sensors (Basel, Switzerland)*, vol. 9, no. 8, pp. 5894–5918, 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3315112/

[22] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-013-9327-2

[23] R. Y. Brogaard, "Control of multi rotor helicopter," 2012.

[24] A. M. Sabatini and V. Genovese, "A stochastic approach to noise modeling for barometric altimeters," *Sensors (ISSN 1424-8220)*, November 2013.

[25] V. Bistrovs and A. Kluga, "Adaptive extended kalman filter for aided inertial navigation system," *Elektronika Ir Elektrotechnika (ISSN 1392 ? 1215)*, 2012.

[26] J. A. Farrell, *Aided Navigation - GPS with High Rate Sensors*. McGraw-Hill Education, 2008.

[27] T. Larsen, N. Andersen, O. Ravn, and N. Poulsen, "Incorporation of time delayed measurements in a discrete-time kalman filter," in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 4, Dec 1998, pp. 3972–3977 vol.4.

[28] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Scituate, MA, USA: Bradford Company, 2004.

[29] N. Gageik, M. Strohmeier, and S. Montenegro, "An autonomous uav with an optical flow sensor for positioning and navigation," *International Journal of Advanced Robotic Systems*, 2013.

[30] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications." in *ICRA*. IEEE, 2013, pp. 1736–1741. [Online]. Available: http://dblp.uni-trier.de/db/conf/icra/icra2013.html#HoneggerMTP13

[31] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, Jun 1994, pp. 935–938.

[32] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 1449–1456.

[33] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 2472–2477.