Implementation of a Multiplicative Extended Kalman filter for Vision-Aided Inertial Navigation based on known world landmarks

TAMU ECEN 662 Fall 2018
Detection and estimation final project

Humberto Ramos

May 6, 2018

1 Introduction

This work is essentially the implementation of a variation of the Extended Kalman filter: a Multiplicative Extended Kalman Filter (MEKF). Here MEKF is used to fuse information coming from a Inertial Measurement Unit (IMU) and a monocular camera in order to estimate the position and attitude (pose) of a rigid body (which carries the sensors) with respect to an inertial frame.

One of the practical reasons to use a Extended Kalman filter to fuse inertial and visual, is that it is possible to have state estimates at high rate (inertial sensor frequency), at the same time that the states estimates are corrected with fairly high precision by feature (from images) measurements, even if this visual inputs are not coming at high frequency. This characteristic of visual-inertial fusion through an EKF is also very attractive for control laws application purposes.

This work is inspired in [1], in which they used a vision-aided inertial navigation system based on a MEKF for the entry,descent and landing of a spacecraft. There, the estimation of the spacecraft pose is estimated with respect to known terrain features.

In this work, the position of the features in the world frame are assumed to be known as well. For this, several arUco targets [2] (which are a special type of marker that facilitates the detection of its features and pose), are used during the experiments.

Furthermore, during the experiments, instead of having a quasi-monotonic approach to the target (as expected for spacecraft landing situations) as in [1], the navigation system used in this application, is moved around the targets to appreciate the quality of the MEKF algorithm results.

In order to validate the state estimates of the MEKF implementation, these are compared against pose information coming from a Motion capture system

(Vicon) which is regarded as the truth. Following an overview of the organization of the implemented system and test the estimation algorithm is presented in section 2, a more in-deep description of the MEKF is presented in section 3. Details about the actual simulation and experiments with real data are shown in section 4 . Section 5 includes brief comments about the limitations of this navigation system given the hardware and configuration used as well as comments comparing this implementation with the results of [1], and finally section 6 provides some conclusions.

2 The experimental framework

In order to actually implement the MEKF for the purposes Visual-Aided inertial navigation, a software/hardware experimental framework was set up. The following is a description of the software, hardware, and arUco targets used for this implementation.

2.1 Open Source Software and data

The open-source robotics framework ROS was used to collect and store data from the IMU, camera, Aruco targets, and Vicon motion capture system. ROS stores the data in a binary format known as a rosbag. The LASR lab library RPM-SAT was used to convert the data from the rosbag into a format that could be sequentially post-processed in Python, while maintaining the original data timing. The open-source library Quatematics provided a Quaternion class that was used to implement equations relating to quaternions. The open source libraries matplotlib and Multiplot2D were used to plot the results. OpenCV was used to perform some operations on the images as well as for visualization of some results.

Quatematics, Multiplot2D, and RPM-SAT are included in the package along with this report. All other libraries mentioned before, must be installed using a package manager. Many additional dependences are required to install all of these libraries, so the best way to ensure that the required libraries are available, is to install the full versions of ROS, OpenCV and Python.

Although the data is first stored and then used by the filter, the data is processed sequentially according to its timestamps, this is, the software gets the measurements at the time when they were actually available at the time of recording the data. Because of the manner this filter is written, it can work online by simply changing just a few lines of code to read the input from a different source, instead of a file. By processing the measurements sequentially and with its actual timestamps, the filter can be tested in a more realistic scenario.

The instructions to be able to run the code for this project, are included in the README file annexed to this project.

2.2 Hardware

An experimental test bed to place the sensors is used in order to acquire real data. This platform was developed at the Land Air and Space Robotics Lab (LASR) at Texas A&M. It consists of a small rigid structure with accurately calibrated holes on it that allow the user to place sensors, at the same time that their relative positions (and sometimes rotations) can be easily measured. The use of this platform, facilitates (i) the acquisition of real data, since it can be attached to a tripod with wheels or easily hand-held, (ii) the measurement of inter sensor parameters, (iii) the measurement of the platform pose with respect to the Vicon reference frame (which is considered to be the truth), since reflective markers (beacons) can be also accommodated on it. Although the test bed can house a variety of elements, for this application we just have a Vectornav VN- 100 IMU (operating at 200 Hz) and a forward-facing Pointgrey BFLY-PGE-13E4C-CS camera.

The camera is a global shutter camera, operating at 20 Hz. Images are recorded at 640x512 pixels. Pixel binning is turned on to increase the signal to noise ratio as the target illumination is poor. Focus and aperture are set manually. Exposure time is 5 ms. This relatively long exposure time reduces the required gain but may not be suitable for fast camera motion. It has a wide field of view (84.7 degrees on this sensor size). This FOV is useful for navigation because features stay view for a "long time." This comes at the cost of distortion which is typically higher for wide FOV lenses. With the purpose of validate the results generated by the estimator, Vicon was utilized to get measurements of the pose of the test-bed. The Vicon system provides inertial measurements for pose with millimeter accuracy at a rate of approximately 100 Hz. The effective Vicon space is limited to a volume of a 3 meters side cube, so the experiments stayed inside this workspace.

As mentioned before, in comparison with [1], the camera is not looking at some terrain to try to detect features on it, instead arUco targets are used as the known references with respect to the world frame.

The features to navigate with respect to, are specifically extracted from four arUco targets. A total of 16 corners positions are available as a result of a feature detection at any instant of time. Because of the design of the arUco targets, their orientation is detected as part of the target detection algorithm. In this way, the 2D and 3D correspondences can always be correctly established, and thus recover the pose appropriately. The libraries used for the corners extraction along with the ROS package arsys were developed by [2] [3] .

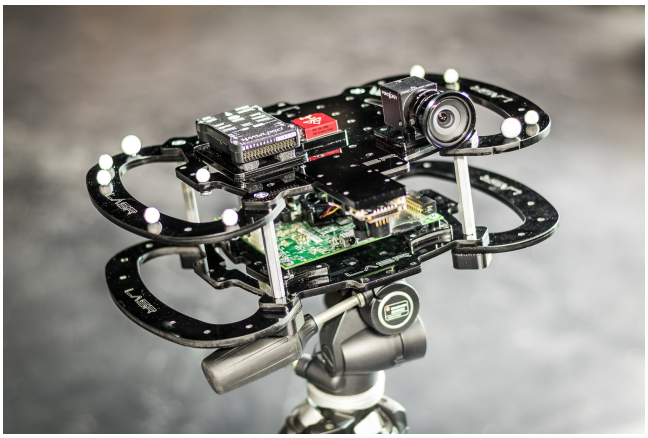

Figure 1: LASR platform in which the sensors are placed to do the experiments. The spherical reflective markers are used by Vicon to track the body.

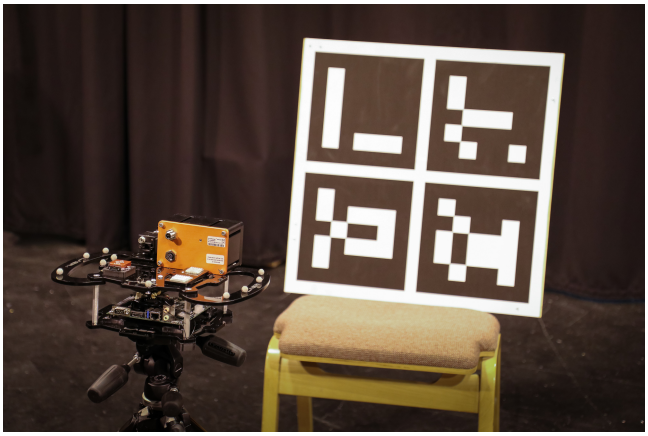

Figure 2: Illustrative picture in how the system is faced towards the arUco target. This picture tries to depict the variety of sensors can be attached to the platform.

2.3 Problem statement

The problem that is solved in this work can be stated as follows: Given IMU and camera observations, estimate the position and attitude of the body that is carrying the sensors, with respect to an inertial frame, as well as the corresponding IMU biases by using a mEKF.

3 The MEKF

3.1 Variable conventions

For this work vectors and matrices are bold characters. Since a vector can be written in any reference frame, the convention used to denote, for example, the position vector $\mathbf{a}$ of the IMU in the world frame, ${}^{W}\mathbf{a}_{I}$ is used. For the attitude that takes a vector from the world frame to the IMU frame, ${}^{I}_{W}\bar{q}$ is written.

3.2 Estimate of prior states

The extended Kalman Filter for this application includes as its states a quaternion (as the parametrization of the attitude), position, velocity, and gyroscope bias and gyroscope accelerometer. This is,

$$\mathbf{x}_I = \begin{bmatrix} {}^{I}_{W}\bar{q}^{\mathrm{T}} & {}^{W}\mathbf{p}_I^{\mathrm{T}} & {}^{W}\mathbf{v}_I^{\mathrm{T}} & \mathbf{b}_g^{\mathrm{T}} & \mathbf{b}_a^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{1}$$

3.2.1 Model assumptions and considerations

The algorithm uses the IMU measurements in order to propagate the states. The kinematics are formulated as a first-order system of differential equations. The derivatives are inertial derivatives. The gyroscope and accelerometer are assumed to be affected by white noise and slow changing drift. The model for the accelerometer and gyro drift are assumed to be zero mean white Gaussian noise.

The gyroscope provides a measurement of the angular velocity $\tilde{\boldsymbol{\omega}}(t)$. $\tilde{\boldsymbol{\omega}}(t)$ relates to the true angular velocity $\boldsymbol{\omega}(t)$ according to the following model:

$$\boldsymbol{\omega}(t) = \tilde{\boldsymbol{\omega}}(t) - \mathbf{b}_g(t) - \mathbf{n}_g(t) \tag{2}$$

$$\dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t) \tag{3}$$

$\mathbf{n}_g(t)$ and $\mathbf{n}_{wg}(t)$ are the gyro measurement noise and drift respectively, both modeled as white Gaussian noise processes.

The accelerometer provides a measurement of the specific force $\tilde{\mathbf{s}}(t)$. $\tilde{\mathbf{s}}(t)$ relates to the true specific force according to the following model:

$$\mathbf{s}(t) = \tilde{\mathbf{s}}(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t) \tag{4}$$

$$\dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \tag{5}$$

$\mathbf{n}_a(t)$ and $\mathbf{n}_{wa}(t)$ are the accelerometer measurement noise and drift respectively, both modeled as white Gaussian noise processes.

3.3 System model

So the kinematic equations are summarized as,

$$ {}^{I}_{I_k}\dot{\bar{q}}(t) = \frac{1}{2}\begin{bmatrix} -\lfloor \boldsymbol{\omega}(t) \times \rfloor & \boldsymbol{\omega}(t) \\ -\boldsymbol{\omega}^{\mathrm{T}}(t) & 0 \end{bmatrix} {}^{I}_{I_k}\bar{q}(t) \tag{6} $$

$$ {}^{W}\dot{\mathbf{p}}_I = {}^{W}\mathbf{v}_I \tag{7} $$

$$ {}^{W}\dot{\mathbf{v}}_I(t) = {}^{W}\mathbf{a}_I(t) = ({}^{I}_{I_k}\mathbf{C}\ {}^{I_k}_{W}\mathbf{C})^{\mathrm{T}}\mathbf{s}(t) + {}^{W}\mathbf{g} \tag{8} $$

$$ = {}^{I_k}_{W}\mathbf{C}^{\mathrm{T}}\ {}^{I}_{I_k}\mathbf{C}^{\mathrm{T}}\mathbf{s}(t) + {}^{W}\mathbf{g} \tag{9} $$

$$ \dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t) \tag{10} $$

$$ \dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \tag{11} $$

Where $\boldsymbol{\omega}(t)$ is the angular velocity of the navigation system in the body or IMU frame, ${}^{W}\mathbf{g}$ is the gravity vector in the world frame, and

$$ \lfloor \boldsymbol{\omega}(t) \times \rfloor \tag{12} $$

is the skew-symmetric matrix of $\boldsymbol{\omega}(t)$.

The MEKF will use the IMU measurements for state prediction. The equations to do this are obtained by computing the expected value from 6 to 11. This is,

$$ \dot{\hat{\mathbf{b}}}_g(t) = \mathbf{0} \rightarrow \hat{\mathbf{b}}_g(t) = \hat{\mathbf{b}}_{g_k} \tag{13} $$

$$ \hat{\boldsymbol{\omega}}(t) = \tilde{\boldsymbol{\omega}}(t) - \hat{\mathbf{b}}_g(t) = \tilde{\boldsymbol{\omega}}(t) - \hat{\mathbf{b}}_{g_k} \tag{14} $$

$$ {}^{I}_{I_k}\dot{\hat{\bar{q}}}(t) = \frac{1}{2}\begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}}(t) \times \rfloor & \hat{\boldsymbol{\omega}}(t) \\ -\hat{\boldsymbol{\omega}}^{\mathrm{T}}(t) & 0 \end{bmatrix} {}^{I}_{I_k}\hat{\bar{q}}(t) \tag{15} $$

$$ \dot{\hat{\mathbf{b}}}_a(t) = \mathbf{0} \rightarrow \hat{\mathbf{b}}_a(t) = \hat{\mathbf{b}}_{a_k} \tag{16} $$

$$ \hat{\mathbf{s}}(t) = \tilde{\mathbf{s}}(t) - \hat{\mathbf{b}}_a(t) = \tilde{\mathbf{s}}(t) - \hat{\mathbf{b}}_{a_k} \tag{17} $$

$$ {}^{W}\dot{\hat{\mathbf{v}}}_I(t) = {}^{I_k}_{W}\hat{\mathbf{C}}^{\mathrm{T}}\ {}^{I}_{I_k}\hat{\mathbf{C}}^{\mathrm{T}}\hat{\mathbf{s}}(t) + {}^{W}\mathbf{g} \tag{18} $$

In order to propagate the equations, Euler integration is used. This will give rise to the expected or predicted or prior belief values. Notice that the biases retain the same value during the computation of the priors.

3.4 Measurement model

The pinhole camera model was used to relate the position of the i^{th} feature in the camera frame (3D coordinates) to the 2D coordinates in the image. That is,

$$\tilde{\mathbf{y}}_{F_i} = \begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \begin{bmatrix} f_x(h_x/h_z) + c_x \\ f_y(h_y/h_z) + c_y \end{bmatrix} + \mathbf{v}_{F_i} \tag{19}$$

where $\mathbf{v}_{F_i}$ is a white noise process with covariance matrix $\mathbf{R}_{F_i} = E[\mathbf{v}\mathbf{v}^{\mathbf{T}}]$ and

$$\begin{bmatrix} h_x & h_y & h_z \end{bmatrix}^{\mathrm{T}} = {}^{C}\mathbf{p}_{F_i} = {}^{C}_{I}\mathbf{C}\left({}^{I}_{W}\mathbf{C}({}^{W}\mathbf{p}_{F_i} - {}^{W}\mathbf{p}_{I}) - {}^{I}\mathbf{p}_{C}\right) \tag{20}$$

where ${}^{C}_{I}\mathbf{C}$ represents the transformation from IMU to camera frame and ${}^{I}\mathbf{p}_{C}$ is the position of the camera in the IMU frame (see figure 3). ${}^{C}_{I}\mathbf{C}$ and ${}^{I}\mathbf{p}_{C}$ are assumed to be known without uncertainty. These values are also refereed as inter-sensor parameters.

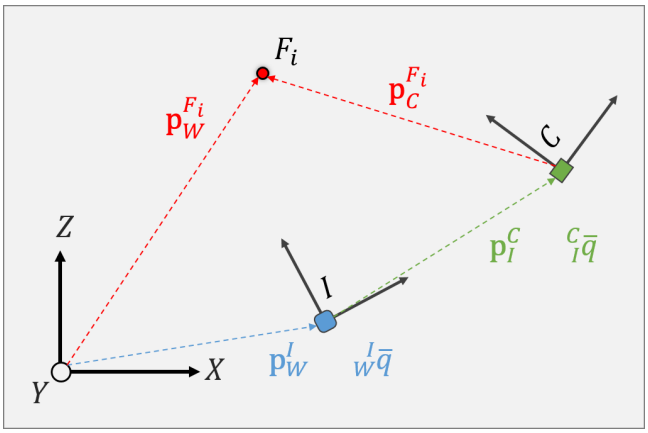

Figure 3: Relationship between World (XYZ), IMU I, camera C and position of the i^{th} feature in the arUco target. The arUco reference frame is considered to be the world frame XYZ. The credit for this figure belongs to [4]

3.5 Measurement Estimate

Similarly as done with the states, it is necessary to have a model for the expected measurements.

$$\begin{bmatrix} \hat{h}_x & \hat{h}_y & \hat{h}_z \end{bmatrix}^{\mathrm{T}} = {}^{C}\hat{\mathbf{p}}_{F_i} = {}^{C}_{I}\mathbf{C}\left({}^{I}_{W}\hat{\mathbf{C}}({}^{W}\hat{\mathbf{p}}_{F_i} - {}^{W}\hat{\mathbf{p}}_{I}) - {}^{I}\mathbf{p}_{C}\right) \tag{21}$$

$$\hat{\mathbf{y}}_{F_i} = \begin{bmatrix} f_x(\hat{h}_x/\hat{h}_z) + c_x \\ f_y(\hat{h}_y/\hat{h}_z) + c_y \end{bmatrix} \tag{22}$$

3.6 Error model and covariance

In order to get the expression to calculate the covariance of the estimator, a expression for the error is needed. In this case, additive error is used for all the quantities except for the quaternion: Multiplicative error is used to describe attitude error. In quaternion algebra this is,

$$ {}^{I}_{W}\bar{q} = \delta\bar{q} \otimes {}^{I}_{W}\hat{\bar{q}} \tag{23} $$

Since the attitude errors are considered to be small, the following approximation is assumed [5].

$$ \delta\bar{q} = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix} \tag{24} $$

$\delta\bar{q}$ is the necessary infinitesimal rotation that aligns the current attitude with the true attitude. It can be seen from 24 that the characterization of the error comes from the three first terms, thus those are the values used to characterize the covariance correspondent to attitude. The continuous-time linearized error dynamics needs to be in the form of

$$ \dot{\check{\mathbf{x}}}_I \approx \mathbf{F}_I \check{\mathbf{x}}_I + \mathbf{G}_I \mathbf{w} \tag{25} $$

And the measurement error in the form,

$$ \check{\mathbf{y}}_k \approx \mathbf{H}\check{\mathbf{x}} + \mathbf{v}_k = \mathbf{H}(\hat{\mathbf{x}}_k)\ \check{\mathbf{x}} + \mathbf{v}_k \tag{26} $$

The $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$ matrices are used in the Kalman filter to approximate the covariance $\mathbf{P}$ of the error states, where

$$ \mathbf{P}(t) = E[\check{\mathbf{x}}(t)\check{\mathbf{x}}^{\mathrm{T}}(t)] \tag{27} $$

The $\dot{\check{\mathbf{x}}}_I$ is the error dynamics state vector. By taking the derivatives with respect to the state error the continuous-time state transition matrix $\mathbf{F}_I$ and the input noise matrix $\mathbf{G}_I$ are obtained

$$ \mathbf{F}_I = \begin{bmatrix} -\lfloor\hat{\boldsymbol{\omega}}\ \times\rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -{}^{I}_{W}\hat{\mathbf{C}}^{\mathrm{T}}\lfloor(\tilde{\mathbf{s}} - \hat{\mathbf{b}}_a)\ \times\rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -{}^{I}_{W}\hat{\mathbf{C}}^{\mathrm{T}} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{28} $$

$$ \mathbf{G}_I = \begin{bmatrix} -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -{}^{I}_{W}\hat{\mathbf{C}}^{\mathrm{T}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{29} $$

$$ \mathbf{w} = \begin{bmatrix} \mathbf{n}_g^{\mathrm{T}} & \mathbf{n}_a^{\mathrm{T}} & \mathbf{n}_{wg}^{\mathrm{T}} & \mathbf{n}_{wa}^{T} \end{bmatrix}^{T} \tag{30} $$

The state error keeps the same structure.

$$\breve{\mathbf{x}}_I = \begin{bmatrix} \delta\boldsymbol{\theta}_I^{\mathrm{T}} & \Delta\mathbf{p}_I^{\mathrm{T}} & \Delta\mathbf{v}_I^{\mathrm{T}} & \Delta\mathbf{b}_g^{\mathrm{T}} & \Delta\mathbf{b}_a^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{31}$$

Finally, regarding with the covariance propagation, we use the common form for the EKF

$$\dot{\mathbf{P}} = \mathbf{F}\mathbf{P} + \mathbf{P}\mathbf{F}^{\mathrm{T}} + \mathbf{G}\mathbf{Q}\mathbf{G}^{\mathrm{T}} \tag{32}$$

Where $\mathbf{Q} = E[\mathbf{w}\mathbf{w}^{\mathbf{T}}]$ is the process noise covariance matrix.

3.7 Measurement error model

For the measurement error, a model is also necessary. The form of the model is

$$\breve{\mathbf{y}}_{F_i} \approx \mathbf{H}_{F_i}\breve{\mathbf{x}} + \mathbf{v}_{F_i} \tag{33}$$

$$\mathbf{H}_{F_i} = \frac{\partial\tilde{\mathbf{y}}_{F_i}}{\partial^C\mathbf{p}_{F_i}} \left[\frac{\partial^C\mathbf{p}_{F_i}}{\partial\breve{\mathbf{x}}_I} \right] \tag{34}$$

$$\frac{\partial\tilde{\mathbf{y}}_{F_i}}{\partial^C\mathbf{p}_{F_i}} = \frac{1}{\hat{h}_z} \begin{bmatrix} f_x & 0 & -f_x\hat{h}_x/\hat{h}_z \\ 0 & f_y & -f_y\hat{h}_y/\hat{h}_z \end{bmatrix} \tag{35}$$

$$\frac{\partial^C\mathbf{p}_{F_i}}{\partial\breve{\mathbf{x}}_I} = {}^C_I\mathbf{C} \begin{bmatrix} \lfloor \left({}^I_W\hat{\mathbf{C}}({}^C\hat{\mathbf{p}}_{F_i} - {}^W\hat{\mathbf{p}}_I) \right) \times \rfloor & -{}^I_W\hat{\mathbf{C}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{36}$$

And finally, the estimate measurement matrix can be formed by blocks (one block per feature) as

$$\mathbf{H}_F = \begin{bmatrix} \mathbf{H}_{F_1} \\ \vdots \\ \mathbf{H}_{F_i} \\ \vdots \\ \mathbf{H}_{F_m} \end{bmatrix} \tag{37}$$

3.8 Extended Kalman Filter Update

Using the measurement model previously described, the state can be updated as is normally done for the EKF. The a priori state $\hat{\mathbf{x}}^-$ and covariance $\mathbf{P}^-$, are the result of the propagation of the kinematic equations. This is, the prior belief. Updates on the states are only performed when a new feature measurement is available according to the following equations [6].

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \tag{38}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_\mathbf{k}\mathbf{H}_k)\mathbf{P}_k^- \tag{39}$$

$$\hat{\breve{\mathbf{x}}}_k^+ = \mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) \tag{40}$$

Then $\hat{\check{\mathbf{x}}}_k^+$ is taken as the correction vector that will be applied to the current state vector. The additive error states are updated using the additive error definition. For example, the estimate ${}^W\hat{\mathbf{p}}_I$ is updated using

$$ {}^W\hat{\mathbf{p}}_I{}^+ = {}^W\hat{\mathbf{p}}_I{}^- + \Delta\mathbf{p}_I^+ \tag{41} $$

The attitude states are updated using the small quaternion error approximation

$$ \bar{q}^+ = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}^+ \\ 1 \end{bmatrix} \otimes \bar{q}^- \tag{42} $$

A brute force normalization is performed for the quaternion after the update. Note that the state error is defined as

$$ \check{\mathbf{x}}_I = \begin{bmatrix} \delta\boldsymbol{\theta}_I^{\mathrm{T}} & \Delta\mathbf{p}_I^{\mathrm{T}} & \Delta\mathbf{v}_I^{\mathrm{T}} & \Delta\mathbf{b}_g^{\mathrm{T}} & \Delta\mathbf{b}_a^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{43} $$

3.9 Additional information on feature extraction and world reference frame

As mentioned before, camera measurements are used to update the states every time features on the arUco target are available. Physically, an arUco target looks like in 4.

3.9.1 Feature extraction

The features to navigate with respect to, are the outer corners extracted from four arUco targets. A total of 16 corners positions are available as a result of a feature detection. Because of the design of the arUco targets, their orientation can also be detected as part of the target detection. In this way, the 2D and 3D correspondences can always be correctly established. More information on arUco target can be found in [2]. OpenCV functions were used to extract the features from the images.

The coordinate reference frame utilized for this work can be seen in figure 5.

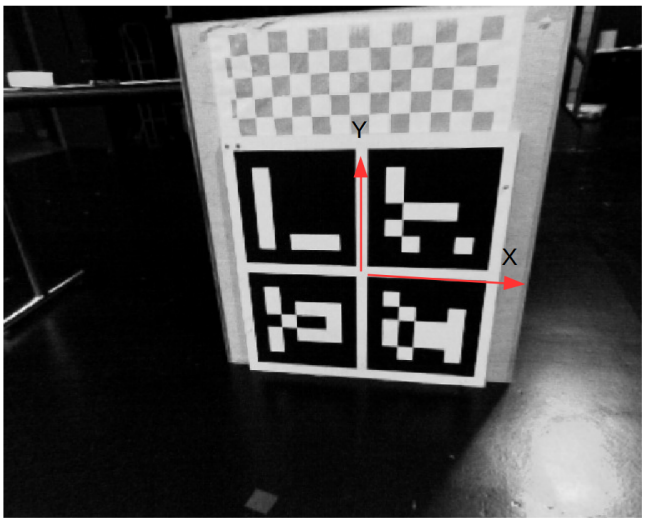

Figure 4: Coordinate world reference frame.

3.9.2 Undistort the feature positions

In order to correct the measurements from radial distortion, instead of correcting the whole image, just the features positions are undistorted. The distortion coefficients were estimated along with the intrinsic parameters of the camera. OpenCV function was used to obtain this parameters, although a own version is also available with the difference that does not force the skew term, neither the focal length to be the same for $\mathbf{f}_x$ and $\mathbf{f}_y$. Once the features have been corrected. They are stored in a measurement vector (to be used later) of the form,

$$\tilde{\mathbf{y}}_F = \begin{bmatrix} \tilde{\mathbf{y}}_{F_1} \\ \vdots \\ \tilde{\mathbf{y}}_{F_i} \\ \vdots \\ \tilde{\mathbf{y}}_{F_m} \end{bmatrix} \tag{44}$$

$$\tilde{\mathbf{y}}_{F_i} = \begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} \tag{45}$$

where $\tilde{u}_i$ and $\tilde{v}_i$ are the pixel coordinates of feature F_i. F_i is located at some position ${}^W\mathbf{p}_{F_i}$.

While the camera matrix is,

$$K = \begin{bmatrix} 432 & 0 & 317 \\ 0 & 432 & 252 \\ 0 & 0 & 1 \end{bmatrix} \tag{46}$$

with radial distortion coefficients

$$k_1, k_2 = [-0.213922, 0.088003] \tag{47}$$

3.9.3 Testing the camera matrix and distortion coefficients

The camera matrix and the distortion parameters were used in one of the images taken with the camera in order to validate them. The OpenCV [7] function to compute the camera matrix and distortion parameters was used, but also the standard algorithm to calibrate the camera was used: a normalized DLT (Direct Linear Transform) followed by a optimization with Levenberg-Marquardt to minimize the re-projection error. The code is included along with this report, but in a separate folder called CameraCalibration. Figure 5 shows the original distorted image, while in figure 6 the undistorted image is seen.

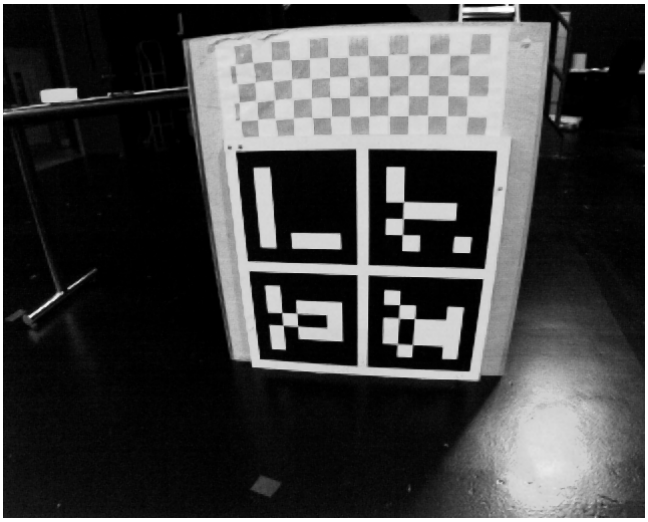

Figure 5: On this image, four arUco targets are shown. Original image. Radial distortion is obvious.

From the undistorted image it can be inferred that the camera parameters are correct. At this point is not known if a more rigorous calibration is required. However, following sections demonstrate this calibration leads to good results. In order to distort the image, just two distortion coefficients were used since according to Zhangs report [8], it is likely that the distortion function is totally dominated by the radial components, and especially by the first term. It also comments, that a more complicated model for the lens distortion could also lead to numerical instability. Thus, the equations that correct the coordinates of the points are

$$\hat{\mathbf{x}} = x + (x - c_x)(k_1 r + k_2 r_2) \tag{48}$$

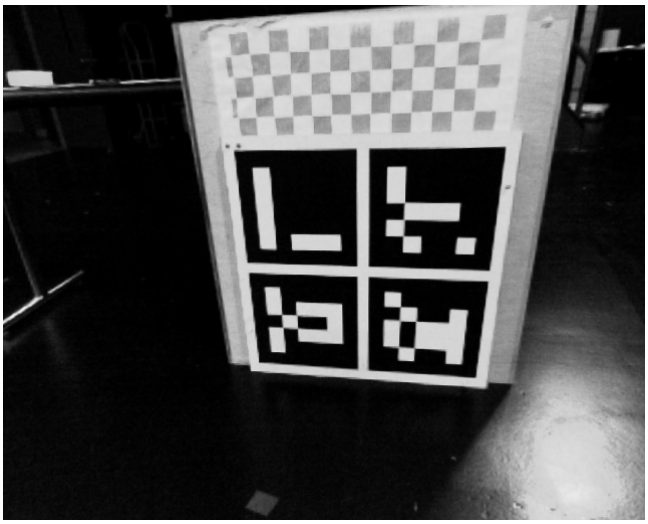

Figure 6: Image after distortion rectification.

$$\hat{\mathbf{y}} = y + (y - cy)(k_1 r + k_2 r_2) \tag{49}$$

Being c_x and c_y the camera center and x and y the original coordinates. The distortion coefficients can also be estimated along with the K matrix in the program which location is mentioned before.

4 Simulations

Either real data (directly from the sensors) or synthetic or both data is necessary to feed the EKF in order to evaluate its performance, as well as to allow and facilitate improvements/debugging. Synthetic data was created based on real data that was smooth and almost with no noise (filtered data).

The way the simulation works allows to use a system clock, such that the measurements are processed as if they were being received in real time. Essentially, a system clock is always running and being compared to the time stamp at which a measurement came. In this case camera and IMU have their own rate, so each of them will be received according to its "true" time, in other words, in the same order and speed they were recorded. This timing treatment, allows one to actually simulate the delays effects expected when running on-line, which simplifies the transition to simulation to real system.

4.1 Validation

A Vicon system is utilized to track the "truth" for the pose of the IMU at the millimeter level. As seen in figure 1, the body has a specific (arrangement) set of reflective markers. This specific configuration is detected by the Vicon system

such that it recognizes the body, and thus can track it. Since the Vicon system has its own reference frame, it is necessary to transform the truth measurements to the corresponding reference frame. In this case, the world reference frame is the arUco target reference frame, thus the attitude of the target is necessary, and reflective markers are also placed on it.

The way the motion capture system 'sees' the two bodies, is shown in figure 8. A closer view of the bodies can be seen in 7.

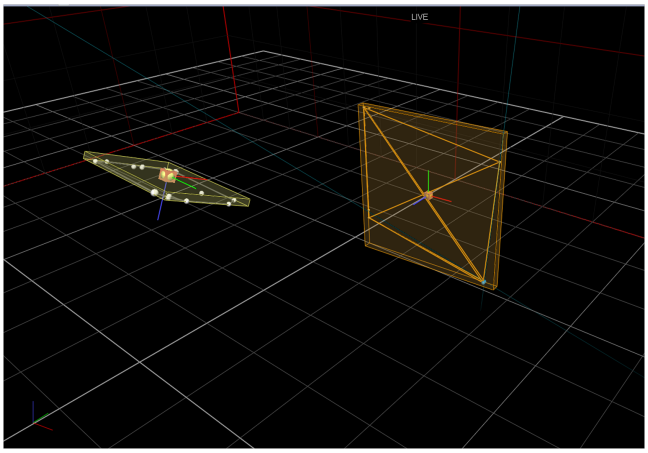

Figure 7: Vicon bodies. The platform where the camera and IMU are (left). The arUco target (right)

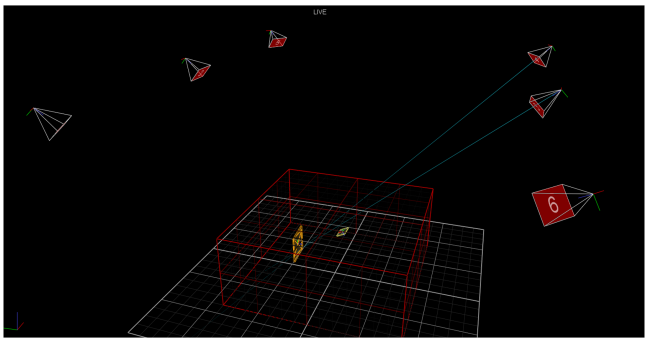

Figure 8: Vicon bodies for the platform and arUco target. The position of the Vicon cameras is also shown.

4.2 Experimental results with real data

4.2.1 Filter simulation and non-uniform time measurements intervals

It is very important to mention that for the synthetic data, the synchronization of the IMU and camera measurements is not a problem, since the time at which the measurements arrive can be artificially set, and the EKF update can be done without problems. However, for the real case (with real data), the IMU and camera measurements arrive to the system at different instants, and the filter could not be applying updates at the proper time if nothing is modified. In order to deal with this problem, the propagation and update stages are done in the following manner. Since the IMU measurement rate (200 Hz) is way higher than the camera rate (20 Hz), is expected that camera measurements come in between IMU measurements. A software routine was designed and implemented to handle this time shifting and propagate and correct the states appropriately with the EKF. Specifically the algorithm that manages the asynchronous measurements can be broadly explained in algorithm 1.

Algorithm 1 States update

Input: IMU and camera measurements
  while IMU measurements coming in propagate equations until the next IMU (Timu (k)) or camera measurement, depending what comes first. **do**
    if Last propagation was performed until the time a camera measurement came Tc **then**
      Time increments are updated so the next propagation is done from the time Tc to the time when the next IMU measurement comes T imu (k+1)
    else
      Propagate IMU measurements
    end if
  end while

4.2.2 Discussion on results

After algorithm 1 was appended to the EKF, better results were obtained compared to what results when synchronism is assumed. The corresponding results are shown in 9,10 and 11. As can be appreciated, when computing the error between the Vicon measurement (considered to be the truth) and the EKF output, the filter seems to be consistent all the time. The errors do not go beyond 6 centimeters for the height, which is the higher peak found for the Z coordinate for this specific trajectory. With respect to the estimated attitude, it is very close to the truth. It can be seen that even for the scale for the quaternion plots, the estimate is almost superimposed with the truth and no bias is observed but a very small offset.This results were obtained after a long process of tuning the Kalman filter. The uncertainties regarding with the camera parameters or IMU were not characterized, none information was had before-hand. So the gains

were adjusted accordingly to the results of each tunning iteration. In general the accuracy of the filter is acceptable.

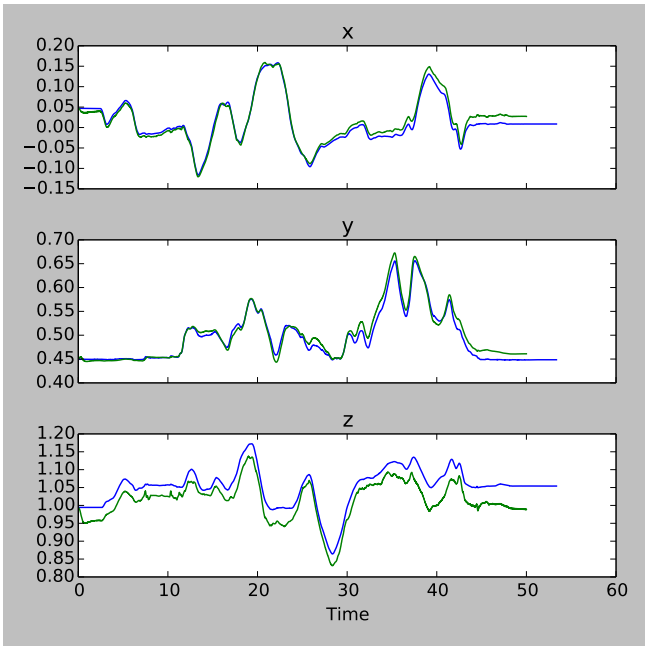

Figure 9: Position in meters of the IMU with respect to the arUco target world frame (in green). The Vicon measurement (in blue)

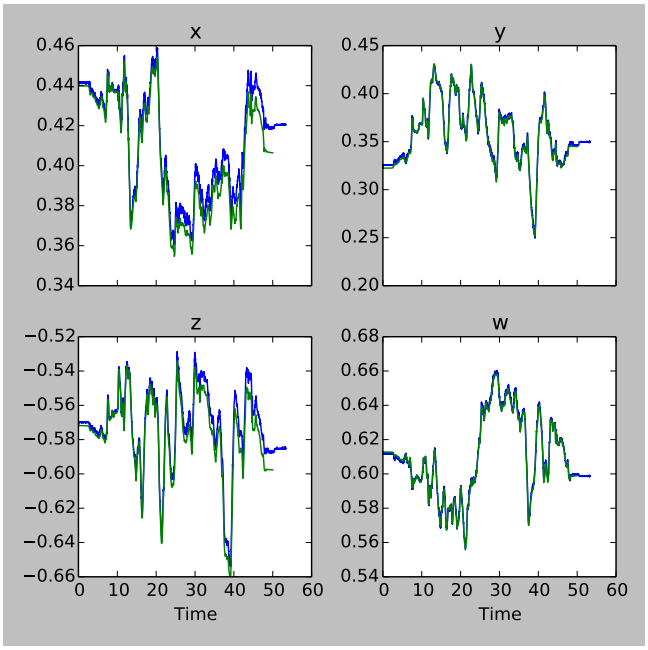

Figure 10: Quaternions for IMU attitude with respect to the arUco target world frame(in green). The Vicon measurement (in blue)

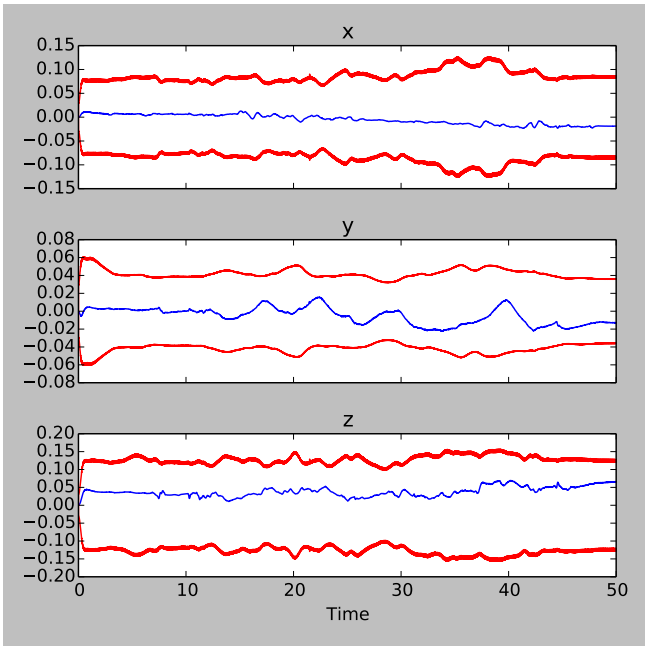

Figure 11: Position errors in meters and 3σ bounds

While iterating for finding better tunning parameters for the filter, it was found that the system is very sensitive to the inter-parameters sensors this is, the position vector and attitude of the IMU with respect to the camera frame. Since these parameters were psychically measured, some iterations to find the better values for the projection center were needed. The best values were selected based on the minimization of the offset generated on the results. The inter-sensor parameters could be estimated along with the states if required.

4.3 Projection of estimated measurements on the image

In order to have a more physical representation of how well the Kalman filter was performing, the estimate of the measurements was projected on the image to be compared with the actual measurements. The closer they are, the more accurate the filter is performing. This is not a rigorous via to evaluate the performance of the EKF implementation, but at least it offers a visual manner to check the "history" of the filter. The projection of the measure estimates are shown in red and the measurements in solid green in figure 12. Note that to save computations, the measurements are the only points undistorted during the execution of the Kalman filter, however for visualization purposes, the whole image is undistorted.

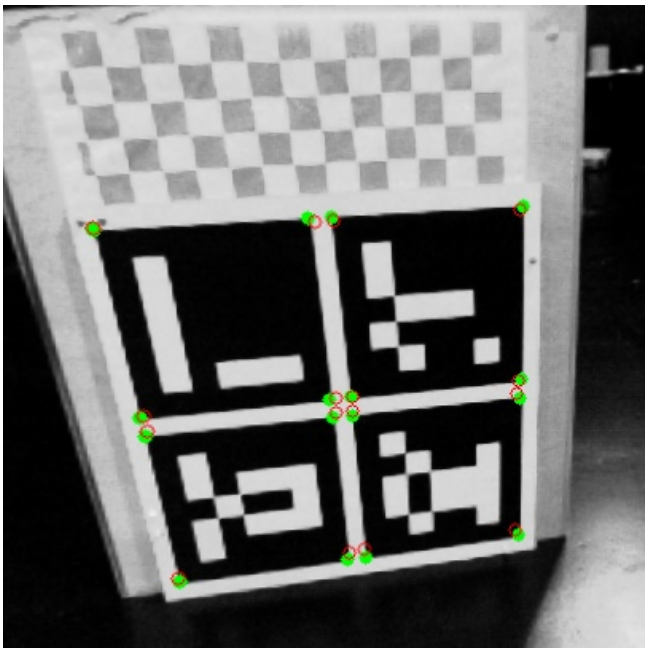

Figure 12: Instantaneous projection of position features estimates (outer corners of the arUco targets, in red) and measurements (in green).

Note that the link between the estimated positions for the features is representative of the performance of the estimated state since the equation to compute the estimated positions for the features is 21. The equation is re-written here for convenience.

$$\begin{bmatrix}\hat{h}_x & \hat{h}_y & \hat{h}_z\end{bmatrix}^{\mathrm{T}} = {}^{C}\hat{\mathbf{p}}_{F_i} = {}^{C}_{I}\mathbf{C}\big({}^{I}_{W}\hat{\mathbf{C}}({}^{W}\hat{\mathbf{p}}_{F_i} - {}^{W}\hat{\mathbf{p}}_{I}) - {}^{I}\mathbf{p}_{C}\big) \tag{50}$$

Thus, if the projected point (computed with 50) is close to the measurement, it is because the estimates ${}^{I}_{W}\hat{\mathbf{C}}$, ${}^{W}\hat{\mathbf{p}}_{F_i}$ and ${}^{W}\hat{\mathbf{p}}_{I}$ are being consistent in general.

5 Considerations and limitations

Even though the filter estimates are very close to the truth states, if the filter does not receive any features for a short period of time, because of the integration errors, biases and noise in the system, the state propagation of the IMU only, will drift indefinitely. This is, in order for this filter implementation to work, features need to be always available. However, even if the features are in the FOV, the possibility of moving too fast such that the image is blurred, can also be a cause of loosing feature measurements temporary.

Also, since the inter-sensor parameters are not estimated but directly measured, if some physical change is made to the system, new tunning parameters

are necessary. This can be solved by augmenting the state vector with the inter-sensor parameters, so they can also be estimated along with the state vector previously defined in 1.

6 Conclusions

Although the implementation of the filter does not include a high number of states, the results are fairly acceptable for navigation purposes when no GPS is available, but just on-board sensors, for example.

This implementation of the MEKF has shown to have a very stable behavior for a set of different trajectories, by always converging in the same way consistently, as long as features to update the states are available. To overcome situations where no enough features are available to update the filter, robustness can be increased by including more sensors, like a LIDAR

Because of the arUco targets have a particular design, the detection of the features is highly robust. As can be seen, in figure 12, even though there is a checker board in the background, the feature detection (corners detection) is not affected at all. If this robustness can be approximated for the case of detecting features in unknown terrain for example, this same implementation could have very interesting results.

The Vicon space is sort of reduce at the LASR Lab, so experiments were limited, but it would be interesting to test this algorithm for a variety of distances with respect to the target.

Finally, since the MEKF implemented uses IMU measurements to propagate the state at 200 Hz and corrects the state estimate (well enough) when features are available, thus estimates for attitude, position and velocity are available at the rate of the IMU, and because of this, they are also very good candidates to be used as an input for motion control.

References

[1] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, "Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks," *Journal of Field Robotics*, vol. 24, no. 5, pp. 357–378, 2007.

[2] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[3] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, 2016.

[4] D. Whitten, "Improving the robustness of monocular vision-aided navigation for multirotos through integrated estimation and guidance," 2017.

[5] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, p. 2005, 2005.

[6] J. L. Crassidis and J. L. Junkins, *Optimal estimation of dynamic systems*. CRC press, 2011.

[7] *The OpenCV Reference Manual*, 2.4.8.0 ed., February 2018.

[8] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.