Evaluation of a lightweight visual–inertial state estimator for GPS-denied environments

Alyssa Ramwell[1*], *Arnold* Pretorius[1], and *Natasha* Botha[2]

[1]MechatronicSystems.Group, Department of Mechanical Engineering, University of Cape Town, 7700, South Africa
[2]Centre for Robotics and Future Production, Council for Scientific and Industrial Research, Pretoria, South Africa.

Abstract Unmanned aerial vehicles are a stimulating and practically promising solution for automating tasks in inaccessible environments. An appealing minimal sensor suite for localisation (the first challenge in automation) is the combination of monocular camera and inertial measurement unit. Although visual localisation is a well-developed field, fiducial marker-less methods remain computationally expensive. In this study, a simple, lightweight visual–inertial state estimator is evaluated in physical experiments, achieving a mean positional and angular error of within 15 cm and 5°, respectively, in non-ideal conditions (periodic image loss and slow update rates). This study's contribution is thus the development, description, and benchmarking of a stripped-down, practical localisation solution that is simple to understand and interface with. After some refinement, the system is expected to serve as a skills development tool and foundation for higher-level navigators.

1 Introduction

Broadly, navigation techniques can be divided into deliberative (global planning, map-based), sensor-based (local planning, mapless), or hybrid approaches [1]. Another distinction lies in their sensor suite. Consumer-grade autonomous unmanned aerial vehicles (UAVs) generally apply a hybrid approach, utilising GPS and global maps for path planning, complemented by a suite of onboard sensors for pose control and height estimation. While GPS-based navigation is a well-tested and widespread solution, it is sensitive to being obscured or disrupted by jamming or in sheltered or complex environments. Further, on its own, it cannot detect height or prevent collisions. Thus, the current research trend in true autonomous navigation is towards sensor-based or hybrid GPS-free solutions [1].

The most general navigation paradigm suitable for complex, unknown environments is simultaneous localisation and mapping (SLAM) [2], where a local map is built up from onboard sensor measurements and maintained over some time horizon, and the robot's position within that map is estimated simultaneously. In a hybrid-style SLAM system, this local map is periodically refined (through loop closures or similar) and incorporated into a global map [1]. While powerful, this approach is computationally expensive and thus

* Corresponding author: alyssa@ramwell.com

inefficient for the many scenarios in which robotic agents operate in relatively static environments. There is therefore a strong argument to be made for GPS-free deliberative navigation paradigms, using known landmarks in pre-mapped areas.

This research focuses on onboard deliberative navigation in GPS-denied environments. More specifically, it considers the first step of such navigation: localisation, i.e., tracking robot pose in an environment.

Localisation approaches can be categorised based on sensor choice and map availability, and these are somewhat inter-related. In terms of sensors for UAV localisation, inertial measurement units (IMUs), combining one or more accelerometers, gyroscopes, and/or magnetometers, are almost ubiquitous in modern robotic systems, with only very niche systems choosing to exclude them. IMUs alone are not sufficient for localisation, however; "minimal" localisation solutions typically combine one or more IMUs with either a GPS, LiDAR, or camera, although more creative solutions exist in the literature. For example, special applications may require the UAV to be equipped with specific sensors for purposes outside of localisation. In such cases, there is incentive to adapt these sensors for localisation purposes as well. Other useful auxiliaries for localisation are barometers, air-flow sensors, and time-of-flight sensors.

The visual–inertial sensor suite, combining one or more cameras and IMUs, is particularly appealing due to its relatively low cost and versatility [3], [4]. As to the latter, visual data is highly information dense and can therefore serve a dual purpose: localisation, and input to higher-level decision makers. However, low-cost cameras have a correspondingly low update rate, making inertial data a useful supplement for tracking high-speed manoeuvres. Many visual–inertial localisation paradigms exist, with the greatest distinction lying in the visual pose estimation algorithm and/or sensor fusion technique.

The most lightweight visual pose estimators rely on fiducial markers, exploiting their planarity to simplify the computation (such as in [5]). In general, this requires the camera to have a relatively close and complete view of the marker. In large environments, this may require hundreds of markers and/or very large markers, which is not always convenient (such as in the case of filming). More general visual pose estimators intend to solve the "perspective-*n*-point" (P*n*P) problem by using *n* (non-planar) feature correspondences. In general, higher numbers of point correspondences translate into greater accuracy, but at the cost of additional computational complexity. Kneip's [6] analytical solution to the P3P problem is widely applied for rapid, relatively computationally-low-cost pose estimation. Although it is often used only as a first step for outlier rejection prior to running more demanding pose solvers [7], evaluations by the authors suggest position errors of less than 20 cm [8].

Of the many existing sensor fusion algorithms, the extended Kalman filter (EKF) is one of the simpler techniques that has been proven effective for UAVs, which exhibit highly non-linear dynamics away from hover conditions. EKFs are highly robust and scalable, having even been applied to relatively small-scale SLAM [2]. For visual–inertial fusion, the inertial navigation system (INS) formulation [9], [10] is common. In this configuration, the state vector tracks the position, orientation (represented as a quaternion), and velocity of the quadcopter. IMU data (total acceleration and angular velocity) are treated as inputs to the EKF and propagated through a dynamics model to predict the quadcopter state at the next timestep. When visual pose estimates become available, these are incorporated as corrections, in the usual EKF formulation.

In this study, we evaluate a localisation system for onboard implementation on a low-cost UAV intended for use in indoor research and as a learning tool. With this application in mind, the localisation solution must be computationally lightweight and sufficiently accurate to prevent collisions in cluttered indoor environments. Ideally, the solution should make use of a sensor suite that is adaptable to alternative, high-level operations that could be pursued in

future research. Finally, this approach should be developed with an emphasis on clarity of implementation and ease of module replacement, to support its pedagogical value.

As such, the proposed localisation system is a loosely-coupled visual–inertial localiser, making use of Kneip's P3P algorithm [6] for visual localisation and fusing this with inertial data using an EKF. This paper records the algorithm development and preliminary evaluation of such a visual–inertial state estimator, as well as a MATLAB Simulink-based environment in which to test it.

The primary contribution of this study is the development, documentation, and benchmarking of a streamlined localisation solution and testing environment that prioritises clarity and ease of integration. While not novel in itself, the system functions as a baseline reference implementation, serving both as a practical platform for skills development and as a foundation for more advanced navigation frameworks.

2 System design

The section below describes the implementation, derivation, and validation of the state estimation algorithms applied in this study, including the simulation environment developed for the latter purpose.

2.1 State estimator

Pose estimation is performed by using an EKF to combine position and orientation estimates dead-reckoned from high-frequency total acceleration and angular velocity measurements with visual pose estimates derived from Kneip's P3P algorithm. The basic system architecture is shown in Figure 1 below.

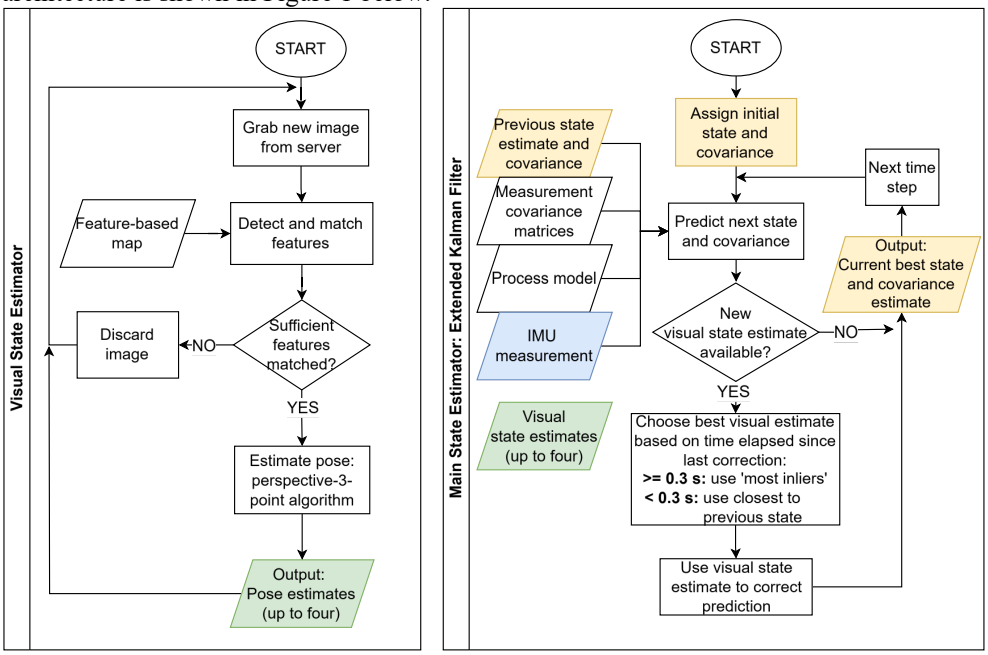

Fig. 1. State estimator architecture.

2.1.1 Visual state estimator

The visual state estimator applied in this study relies on Kneip et al.'s [6] solution to the P3P problem. Although the system aims for marker-less state estimation, the evaluations reported herein are based on point correspondences obtained by feature matching using an in-scene checkerboard and MATLAB's built-in detector based on [12].

The general formulation of the P3P problem is derived from simple trigonometry and the pinhole camera model. The pinhole model assumes that light from the environment passes through an infinitesimal aperture (the centre of projection, C) without bending and is projected onto a sensor plane some distance (focal length f) away, as shown in Figure 2. This is often simplified using congruent triangles to visualise the image plane in front of the aperture. Notice that the real-world points also lie somewhere along these vector lines.

Assume that there are three distinct landmark points in an environment and that their precise locations relative to some world coordinate system $\{W\}$ are known. Also assume that the points can be uniquely identified in a digital photograph of the environment (i.e., their pixel co-ordinates in the image are known). If the pixel densities of the camera sensor ($\lambda_x$and λ_y) and the focal length (f) are known, i.e., after the camera is calibrated, the 3D co-ordinates of these points on the virtual image plane can be identified.

From these image plane coordinates, the angles between the light vectors (reflecting off those points) can be calculated. The distances between the world points are known, the law of cosines can obtain three (quadratic) expressions for the lengths of the vectors between the camera aperture and the world points. This system of equations can be solved analytically (non-uniquely) but is computationally inefficient and poorly conditioned owing to the nature of the variables and operations involved.

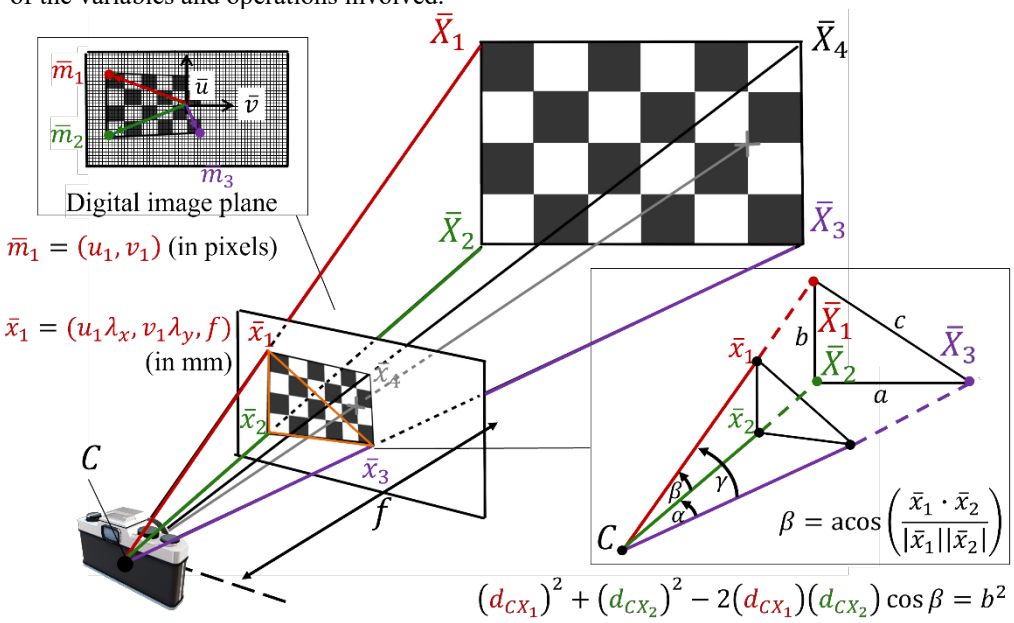

Fig. 2. Summary of general P3P problem formulation based on the pinhole camera model.

Instead, Kneip et al. [6] introduce a few ingenious intermediate reference frames which enable the quartic closed-form solution to be expressed in terms of a single cosine variable ($\cos\theta$, where θ is the angle between one newly defined axis η_y and the plane X_1X_2C – see Figure 3), which is more computationally stable than the distance variable from the general

solution. The original MATLAB implementation of the algorithm is used, and can be found at [11].

In both cases, the output is up to four potential solutions (poses), which must be disambiguated. Commonly, disambiguation is done using additional point correspondences, as follows. For each additional point correspondence, the reprojection error resulting from each potential pose solution is calculated. Following this, the best solution is chosen as that which produced the minimum mean reprojection error or the most inliers (the most reprojection errors within a prespecified limit). Alternatively, if a good, recent estimate of the camera's pose is available, this can be used to infer the most likely solution (or narrow down the number of potential solutions).

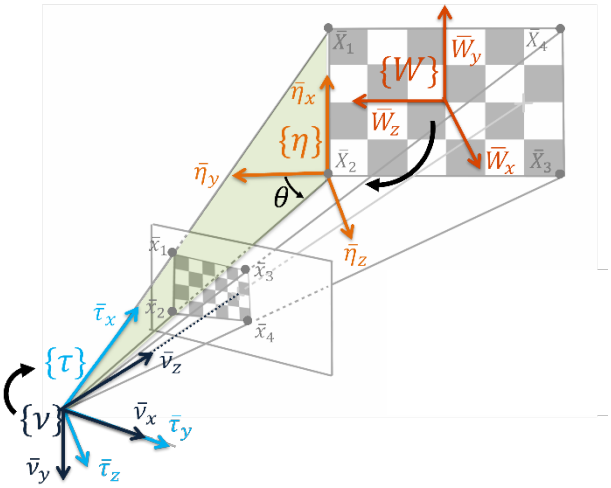

Fig. 3. Intermediate reference frames introduced by Kneip et al. [6].

In the proposed system, these two methods ("most inliers" and "smallest change", respectively) are combined. In general, the "smallest change" method is applied. However, when this becomes unreliable (in cases where the effective camera frame rate is low and integration errors build up between corrections), "most inliers" is applied, to decouple the disambiguation from the error in the current state estimate.

2.1.2 Extended Kalman filter

In this study, we apply an additive full-state EKF, with the IMU measurements formulated as pseudo control inputs to the process model, and P3P results formulated as corrective measurements. Since, in general, IMUs have a much higher update rate than cameras, the EKF is configured to operate asynchronously, with IMU-based state predictions made every iteration, and visual data only incorporated when it becomes available.

The EKF attempts to keep track of the full state of the quadcopter ($\mathbf{x}$) and its associated covariance $\mathbf{P}$ over discrete time. At its minimum, the quadcopter state vector is made up of the position ${}_W\mathbf{p}_B$, orientation ${}_W\mathbf{q}_B$ (in this case, as a quaternion), and linear velocity ${}_W\mathbf{v}_B$ of the quadcopter body frame $\{B\}$ in the world frame $\{W\}$ (in this paper, the bottom right subscript identifies the body to which the variable applies, while the bottom left subscript indicates the reference frame in which the variable is described). We also evaluate a version of the EKF where the state vector is augmented with ${}_I\mathbf{b}_a$ and ${}_I\mathbf{b}_g$, the IMU's accelerometer

(subscript a) and gyroscope (subscript g) biases in the IMU frame $\{I\}$. Thus, the complete state vector at time *k* is:

$$\mathbf{x}_k = \left[{}_W\mathbf{p}_B^{\mathrm{T}} \; {}_W\mathbf{q}_B^{\mathrm{T}} \; {}_W\mathbf{v}_B^{\mathrm{T}} \; {}_I\mathbf{b}_{\boldsymbol{a}}^{\mathrm{T}} \; {}_I\mathbf{b}_{\boldsymbol{g}}^{\mathrm{T}} \right]_k^{\mathrm{T}}. \tag{1}$$

What comes out of the EKF is not the next true state, however, but rather an estimate thereof, which is updated every iteration. In fact, two new estimates are generated each iteration: one predictive or *a priori* estimate, and one corrected or *a posteriori* estimate. To differentiate, a priori estimates are indicated with a hat (e.g., $\hat{\mathbf{x}}_k$), a posteriori estimates are indicated with an additional '+' superscript (e.g., $\hat{\mathbf{x}}_k^+$), and true values have neither (e.g. $\mathbf{x}_k$).

In the prediction phase, the next (*k+1*) a priori state estimate is obtained by propagating the current (*k*) a posteriori estimate through a process model (a.k.a. state transition model or dynamics model) incorporating any actions known to have been taken by the quadcopter at this time (control inputs). In our case, these actions are the IMU measurements, treated as pseudo-control inputs. They are denoted by the control input vector $\mathbf{u}$, comprising the measured total acceleration (including gravity) and angular velocity of the IMU in the IMU frame, denoted ${}_I\mathbf{u}_a$ and ${}_I\mathbf{u}_g^{\mathrm{T}}$, respectively:

$$\mathbf{u}_k = \left[\; {}_I\mathbf{u}_g^{\mathrm{T}} \; {}_I\mathbf{u}_a^{\mathrm{T}} \; \right]_k^{\mathrm{T}}. \tag{2}$$

The IMU is subject to noise and biases, which render its measurements imperfect. This is captured by the IMU measurement model given below, where ${}_I\mathbf{a}_{I_k}$ and ${}_I\boldsymbol{\omega}_{I_k}$ are the ideal measured acceleration and angular velocity of the IMU in the IMU frame at time *k*:

$$\begin{bmatrix} {}_I\boldsymbol{\omega}_I \\ {}_I\mathbf{a}_I \end{bmatrix}_k = \begin{bmatrix} {}_I\mathbf{u}_{g_k} - \; {}_I\mathbf{b}_{g_k} - \mathbf{n}_{g_k} \\ {}_I\mathbf{u}_{a_k} - \; {}_I\mathbf{b}_{a_k} - \mathbf{n}_{a_k} \end{bmatrix} \tag{3}$$

Here, ${}_I\mathbf{b}_{g_k}$and ${}_I\mathbf{b}_{a_k}$, and $\mathbf{n}_{a_k}$ and $\mathbf{n}_{a_k}$ represent the biases and noise experienced by the gyroscope and accelerometer, respectively, at time *k*. Inherent to the EKF is the assumption that noise can be modelled as a zero-mean gaussian: $\mathbf{n}_i \sim N(0, \boldsymbol{\sigma}_i^2)$. Thus, whenever Eq. (3) is evaluated, it is evaluated with $\mathbf{n}_i = \bar{\mathbf{n}}_i = \mathbf{0}$, and the effect of noise is rather captured by considering the variance (and covariance) of the measurements, captured in the process noise covariance matrix $\mathbf{Q}$, which is defined in the noise space in this study. Finally, note that the accelerometer measures specific force, which is equal to acceleration once the gravity vector is removed – this effect is captured later.

Back to the prediction: although a six-degree-of-freedom quadcopter is a complex, non-linear system, its state dynamics can be adequately approximated using linearization (e.g., a first-order Taylor expansion) or numerically integrated using methods such as the trapezoidal rule. Thus, the state update equation can be considered a function of the two previous a posteriori state estimates and IMU measurements:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \Delta t \cdot \dot{\hat{\mathbf{x}}}_k = f(\hat{\mathbf{x}}_k^+, \hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_k, \mathbf{u}_{k-1}) \tag{4}$$

$$\hat{\mathbf{x}}_k = \left[{}_W\hat{\mathbf{p}}_B^{\mathrm{T}} \; {}_W\hat{\mathbf{q}}_B^{\mathrm{T}} \; {}_W\hat{\mathbf{v}}_B^{\mathrm{T}} \; {}_I\hat{\mathbf{b}}_{\boldsymbol{a}}^{\mathrm{T}} \; {}_I\hat{\mathbf{b}}_{\boldsymbol{g}}^{\mathrm{T}} \right]_k^{\mathrm{T}}. \tag{5}$$

At the first instance, since the EKF will not yet have run, both the initial a posteriori estimated state $\hat{\mathbf{x}}_0^+$ and state covariance $\hat{\mathbf{P}}_0^+$ must be initialised: we assume the quadcopter is released from rest in a known pose with a known uncertainty.

To evaluate Eq. (2), we need to find $f(\hat{\mathbf{x}}_k^+, \hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_k, \mathbf{u}_{k-1})$. The position update is simply a function of the previous position and the average of the two previous velocities:

$$\hat{\mathbf{p}}_{k+1} = {}_W\hat{\mathbf{p}}_{B_k}^+ + \frac{\Delta \mathrm{t}}{2}\left({}_W\hat{\mathbf{p}}_{B_k}^+ + {}_W\hat{\mathbf{p}}_{B_{k-1}}^+ \right) = {}_W\hat{\mathbf{p}}_{B_k}^+ + \frac{\Delta \mathrm{t}}{2}\left({}_W\hat{\mathbf{v}}_{B_k}^+ + {}_W\hat{\mathbf{v}}_{B_{k-1}}^+ \right). \tag{6}$$

The orientation vector is a function of the previous orientation and the trapezoidal integration of the two previously measured angular velocities (from the IMU). Firstly, the trapezoidal relationship is approximated by

$$ {}_I\bar{\mathbf{u}}_{g_k} = \frac{{}_I\mathbf{u}_{g_k} + {}_I\mathbf{u}_{g_{k-1}}}{2}. \tag{7} $$

There is a problem: this is a 3D vector in the IMU frame, while the quadcopter orientation is tracked as a quaternion in the inertial reference frame. Further, it would be preferable to use the *true* or *ideal* angular velocity in the update, rather than the imperfect measurement. Thus, the average angular velocity measurement must first be manipulated to compensate for bias and reference frame mismatch, and then augmented into a pure quaternion (note that translation between the IMU and quadcopter is assumed to be negligible)

$$ {}_B\boldsymbol{\Omega}_{B_k} = \begin{bmatrix} 0 \\ {}_B\boldsymbol{\omega}_B \end{bmatrix}_k = \begin{bmatrix} 0 \\ {}_B\mathbf{R}_I \cdot \left({}_I\bar{\mathbf{u}}_g - {}_I\hat{\mathbf{b}}^+_{g_k} - \mathbf{n}_g \right) \end{bmatrix}_k \tag{8} $$

Thus, the update equation for the orientation quaternion is

$$ \hat{\mathbf{q}}_{k+1} = {}_W\hat{\mathbf{q}}^+_{B_k} + \frac{\Delta t}{2} \left[{}_W\hat{\mathbf{q}}^+_{B_k} * {}_B\boldsymbol{\Omega}_{B_k} \right]. \tag{9} $$

These equations do not maintain the quaternion unit norm, so the new quaternion estimate must be normalised – the normalisation equation is excluded from the later Jacobians.

The velocity update equation must similarly account for reference frame mismatches, as well as the gravity vector ${}_W\mathbf{g}$:

$$ \hat{\mathbf{v}}_{k+1} = {}_W\hat{\mathbf{v}}^+_{B_k} + \frac{\Delta t}{2} \cdot \left({}_W\hat{\mathbf{v}}_{B_k} + {}_W\hat{\mathbf{v}}_{B_{k-1}} \right), \tag{10} $$

where

$$ {}_W\hat{\mathbf{v}}_{B_k} = {}_W\hat{\mathbf{R}}^+_B \cdot {}_B\mathbf{R}_I \left({}_I\bar{\mathbf{u}}_a - {}_I\hat{\mathbf{b}}^+_{a_k} - \mathbf{n}_a \right) + {}_W\mathbf{g}. \tag{11} $$

Finally, although the bias terms may be time-varying, they cannot be measured by the sensor in which they present. Thus, in the prediction stage, they are only affected by noise:

$$ \begin{bmatrix} {}_I\hat{\mathbf{b}}_a \\ {}_I\hat{\mathbf{b}}_g \end{bmatrix}_{k+1} = \begin{bmatrix} {}_I\hat{\mathbf{b}}^+_{a_k} - \mathbf{n}_{ba_k} \\ {}_I\hat{\mathbf{b}}^+_{g_k} - \mathbf{n}_{bg_k} \end{bmatrix}. \tag{12} $$

Equations (6)–(12) represent the full process model. At each timestep, the previous measurements and state estimates are substituted into these equations to get the next state prediction. The predicted state covariance $\hat{\mathbf{P}}_{k+1}$ must also be calculated based on

$$ \hat{\mathbf{P}}_{k+1} = \mathbf{F}_k \hat{\mathbf{P}}^+_k \mathbf{F}^{\mathrm{T}}_k + \mathbf{L}_k \mathbf{Q}_k \mathbf{L}^{\mathrm{T}}_k. \tag{13} $$

Here, $\hat{\mathbf{P}}_{k+1}$ is the next estimated state covariance, $\mathbf{Q}_k$ is the current process noise covariance in the noise space, and $\mathbf{F}_k$ and $\mathbf{L}_k$ are the current Jacobians of the process model (Eq. 4) with regards to the state vector $\mathbf{x}$ and process noise vector $\mathbf{n}_\mathbf{Q}$, respectively. The process noise vector is

$$ \mathbf{n}_Q = \left[\mathbf{n}^{\mathrm{T}}_g \; \mathbf{n}^{\mathrm{T}}_a \; \mathbf{n}^{\mathrm{T}}_{ba} \; \mathbf{n}^{\mathrm{T}}_{bg} \right]^{\mathrm{T}}. $$

Thus, the next state prediction and its uncertainty are obtained from previous states and IMU measurements. If no corrective measurements are received, the a priori estimates are considered the current best estimates and assigned as the a posteriori estimates as well:

$$ \hat{\mathbf{x}}^+_{k+1} = \hat{\mathbf{x}}_{k+1}, \tag{14} $$

$$ \hat{\mathbf{P}}^+_{k+1} = \hat{\mathbf{P}}_{k+1}. \tag{15} $$

This dead-reckoning approach is repeated until visual localisation data is received, in which case the EKF's correction step is called to refine the estimate using the new camera measurement.

The P3P-based visual pose estimator implemented in this study outputs the estimated pose of the camera in the world frame. The camera measurement vector $\mathbf{z}$ at sample instance *k* is:

$$ \mathbf{z}_k = \begin{bmatrix} {}_W\mathbf{p}_C + \mathbf{n}_{pc} \\ {}_W\mathbf{q}_C + \mathbf{n}_{qc} \end{bmatrix}_k, \tag{16} $$

where $\mathbf{n}_{pc_k}$ and $\mathbf{n}_{qc_k}$represent noise in the position and orientation quaternion estimates, respectively, from the visual state estimator. The associated measurement noise covariance

is denoted **V**. The measurement model, reflecting the estimated measurement as a function of the a priori state estimate, is given by

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k) = \begin{bmatrix} {}_W\hat{\mathbf{p}}_{B_k} + {}_W\hat{\mathbf{R}}_{B_k}\, {}_B\mathbf{p}_C \\ {}_W\hat{\mathbf{q}}_{B_k} * {}_B\mathbf{q}_C \end{bmatrix}. \tag{17}$$

Here, ${}_W\hat{\mathbf{R}}_{B_k}$ is the estimated rotation matrix from {B} to {W} at instance *k*, compiled from the corresponding quaternion in the estimated state vector; ${}_B\mathbf{p}_C$ and ${}_B\mathbf{q}_C$ are the constant position and orientation of the camera frame $\{C\}$ with respect to $\{B\}$. The asterisk symbol $(*)$ represents the Hamilton product.

In the correction stage, the obtained measurement is compared to the predicted measurement. The difference, the measurement residual, is weighted by a matrix gain called the Kalman gain and added to the predicted state to adjust it towards the corrective measurement.

The first step is to receive the corrective measurement $\mathbf{z}_{k+1}$. The next is to calculate the predicted measurement $\hat{\mathbf{z}}_{k+1}$, from (17), using the most recent a posteriori state estimate $\hat{\mathbf{x}}_{k+1}$. The predicted measurement covariance $\hat{\mathbf{S}}_{k+1}$ is obtained from

$$\hat{\mathbf{S}}_{k+1} = \mathbf{H}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^{\mathrm{T}} + \mathbf{V}_{k+1}, \tag{18}$$

where $\mathbf{H}_{k+1}$ is the jacobian of the measurement model with respect to the state vector, evaluated at $\mathbf{x} = \hat{\mathbf{x}}_{k+1}$.

The measurement residual $\mathbf{y}_{k+1}$ is calculated using

$$\mathbf{y}_{k+1} = \mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1}, \tag{19}$$

and the Kalman gain $\mathbf{K}_{k+1}$, used to weight the correction, is obtained from

$$\mathbf{K}_{k+1} = \hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^{\mathrm{T}}\hat{\mathbf{S}}_{k+1}^{-1}. \tag{20}$$

The corrected (a posteriori) state estimate is then simply the sum (hence, the "additive" EKF naming) of the predicted state and the Kalman gain-weighted measurement residual. This is given by

$$\hat{\mathbf{x}}_{k+1}^{+} = \hat{\mathbf{x}}_{k+1} + \mathbf{K}_{k+1}\mathbf{y}_{k+1} \tag{21}$$

with the corresponding error covariance obtained using

$$\hat{\mathbf{P}}_{k+1}^{+} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\hat{\mathbf{P}}_{k+1}. \tag{22}$$

2.2 Simulation environment

A secondary goal of this study is the development and validation of a simulation environment that reflects realistic hardware limitations, which enables rapid software-in-the-loop testing and experimentation. A simulation environment was developed in MATLAB Simulink for use in validation and further experimentation. The simulation environment makes use of a range of built-in hardware models, including a camera, a UAV, an IMU, and a 3D visualisation environment based on Unreal Engine. The IMU block models a range of calibration and noise parameters; in this study, we use only velocity random walk, rate random walk, and bias instability for the accelerometer and gyroscope. For the camera model, we vary the typical intrinsics (focal length and principal point), resolution, and radial and tangential distortion factors.

The main contribution of this study to the simulation environment is the development of various mapping set-up scripts, coordinate transformations, and state calculations required to convert trajectories defined in a right-handed world frame into appropriate signals for the quadcopter, IMU, and camera blocks. The original model is available at github.com/ARamwell/QuadSimEnv/releases/tag/robmech-2025.

2.3 State estimator validation

The visual state estimator and EKF were validated using simulated data from an ideal IMU (1000 Hz data rate, all noise and bias parameters set to zero) and camera (no distortion, arbitrarily high resolution). When applying a dead-reckoning approach (no camera corrections), the maximum position and orientation errors obtained for an arbitrary reference trajectory over 16 seconds were 0.06 m and 0.05°, respectively. The position tracking result for the complete 16-state EKF with trapezoidal integration is shown in Fig. 4, with an example of the evolution of the world frame z-position shown in Fig. 5.

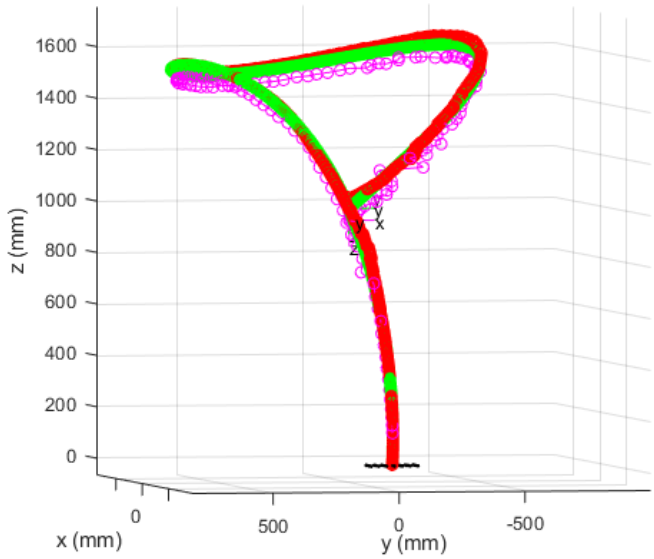

Fig. 4. Position tracking result of 16-element EKF using trapezoidal integration with ideal IMU and camera data, with the EKF estimate shown in red, the visual state estimate (of the camera) in pink, and the true position in green.

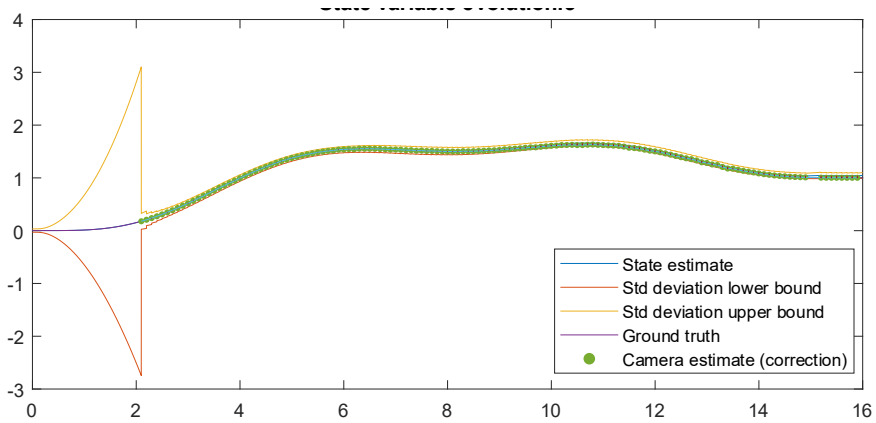

Fig. 5. State variable evolution: z-position. Standard deviations, extracted from the EKF covariance matrix, are also shown.

These results demonstrate that the EKF is working as expected: new measurements (in green) reduce the uncertainty in the estimate, which otherwise grows unbounded (as indicated by the standard deviation bounds). With perfect data, the EKF tracks the pose near-perfectly,

with the small position and orientation errors in the dead-reckoning EKF attributed to integration errors. This validates the process model.

3 Test methods

This section details the physical tests conducted to evaluate the proposed state estimator, which also involved calibrating and characterising the physical sensors.

3.1 Performance evaluation method

The pose estimation accuracy of the following variants of the proposed state estimator were evaluated on hardware over several arbitrary trajectories:

1. 10-state EKF (excluding sensor bias) with trapezoidal integration
2. 10-state EKF (excluding sensor bias) with first-order (rectangular) integration
3. 16-state EKF (including sensor bias) with trapezoidal integration
4. 16-state EKF (including sensor bias) with first-order (rectangular) integration

The specific evaluation metrics are mean orientation and position error. Position error is calculated as the Euclidean distance between the estimated and actual position. Orientation error is calculated as the absolute value of α, the minimum rotation angle between the estimated and actual quaternion orientation, as follows:

$$ {}_W\hat{\mathbf{q}}_{B_k}^{+} * \left({}_W\mathbf{q}_{B_k}\right)^{-1} = \begin{bmatrix} \cos\left(\frac{\alpha}{2}\right) \\ \mathbf{v} \cdot \cos\left(\frac{\alpha}{2}\right) \end{bmatrix} \tag{23} $$

$$ \epsilon_{q_k^+} = |\alpha| \tag{24} $$

In simulation, the true pose is known; in physical tests, the ground truth is obtained from the motion capture system. The combined pose error, when used, is defined as the sum of the position and orientation errors, with $\epsilon_{q_k^+}$ in radians.

The pose error is calculated at each timestep for each trial and estimator variant. The trials are combined based on the time elapsed since system initialisation, with measurements from different trials being binned into 0.1s intervals. The mean error in each bin is calculated, and the estimators are then compared according to the means of their bin errors.

3.2 Hardware

The proposed visual–inertial state estimator is intended for online execution, entirely onboard a UAV. However, for the early physical evaluations reported on in this study, the sensor fusion and image processing algorithms are implemented offboard and offline in MATLAB and Simulink, and hardware is used purely to obtain realistic measurements.

Fig. 6 depicts the hardware and data capture architecture used in the physical experiments of this study. A Pixhawk 6C flight controller [13] and ESP32-CAM are mounted on a base (“fakeDrone”). The flight controller houses the IMU, runs PX4 firmware, and uses the uXRCE-DDS middleware to exchange accelerometer and gyroscope data in a ROS2-compatible format. Uncalibrated IMU readings are transmitted via 900 MHz telemetry to the uXRCE-DDS agent on the host computer, where it is published to a ROS2 topic and logged to and timestamped on arrival by a ROS2 listener implemented in MATLAB. The ESP32-CAM camera is mounted underneath the flight controller, as would be the case in a real quadcopter. The camera uses WiFi to stream images to a simple web server at a rate of between 6-10 frames per second. The server is polled iteratively by a MATLAB script running in parallel with the IMU listener. New images are timestamped on arrival and saved

to disk. An Optitrack motion capture system (consisting of eight Prime^X 13 cameras [14], calibrated on the testing day) tracks the orientation and position of the fakeDrone assembly, which is fitted with four reflective markers on an external rigid body base, attached to the assembly using reusable putty adhesive.

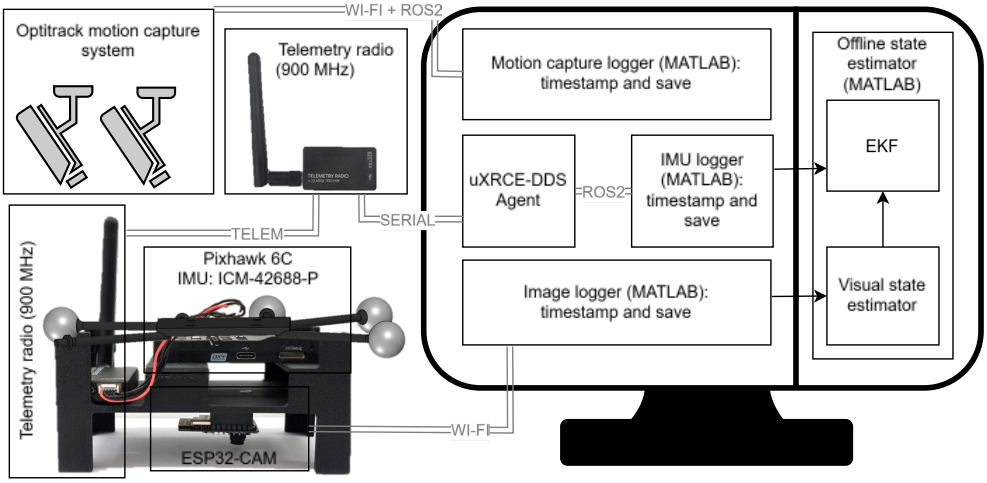

Fig. 6. Hardware architecture for physical tests

3.3 Camera calibration

The camera used in this study was calibrated using MATLAB's built-in camera calibration application. The intrinsic matrix obtained for the ESP32-CAM at SVGA resolution was estimated to be

$$\mathbf{K}_c = \begin{bmatrix} 458.94 & 0 & 249.58 \\ 0 & 459.54 & 172.63 \\ 0 & 0 & 1 \end{bmatrix}. \quad (25)$$

Further parameters used to simulate the camera (but not applied in the visual state estimator's camera model) are the radial and tangential distortion coefficients:

$$\mathbf{K}_r = [-0.0438 \quad 0.1349] \quad (26)$$

$$\mathbf{K}_t = [-0.0051 \quad -0.0039]. \quad (27)$$

3.4 P3P error characterisation

In these tests, we aim to answer the question: how good is the best of the four P3P solutions? The results are used to populate the camera measurement covariance for the EKF and to inform future application constraints, such as feature size and range requirements for accurate localisation.

The accuracy of the visual state estimator was evaluated based on the data obtained in the physical tests described above. The camera was tracked (by the motion capture system) as it moved around an area of approximately 4 × 4 × 2 m^3 (l × w × h), capturing images of the checkerboard target at various distances and orientations. The P3P algorithm is applied to each image, resulting in up to four potential pose estimates. The estimate is evaluated on the best of those four solutions, as determined by the minimum pose error compared to the ground truth (from motion capture data).

Fig. 7 shows the results of the visual state estimator evaluations. In prior tests, it was noted that there is relationship between the position of the camera relative to the target and

the accuracy of the P3P position estimate, as evidenced below and in the summary in **Table 1**. This may be dependent on the spacing of detected features and should be investigated in future work.

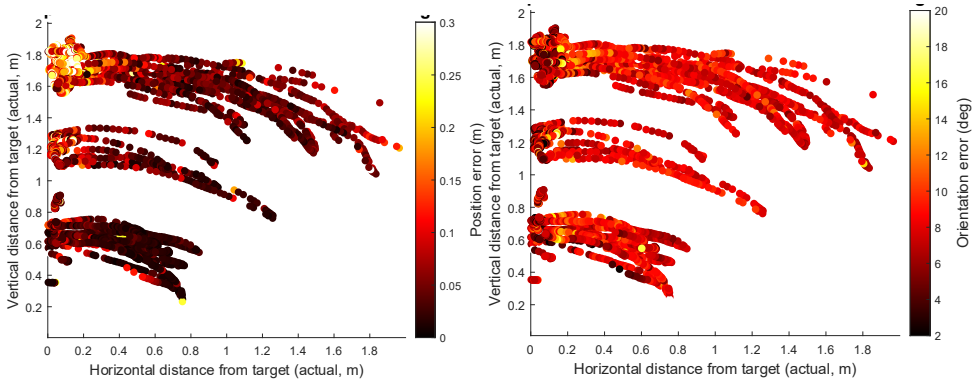

Fig. 7. Visual state estimator error as a function of vertical and horizontal distance from target (checkerboard): (a) position estimate error, (b) orientation estimate error.

Table 1. Pose estimate error of the visual state estimator, evaluated in physical trials

Data group (grouped by z-position)	Position error (m)			Orientation error (deg)		
	Mean	Std dev	$\mu + \sigma$	Mean	Std dev	$\mu + \sigma$
High	0.1204	0.0863	0.2067	7.4138	2.8424	10.2562
Med	0.0786	0.0608	0.1394	6.8450	4.7811	11.6261
Low	0.0321	0.0350	0.0671	8.0454	9.6630	17.7084
Combined	0.0758	0.0748	0.1506	7.5248	6.7263	14.2511

3.5 Camera measurement covariance

In this loosely-coupled filtering approach, the camera measurement noise is characterised in the state space (i.e., in terms of positional and angular error). While pixel noise may be Gaussian, it does not map to a Gaussian relationship in the state space. Although the EKF is predicated on the assumption that system noise is Gaussian, it is often applied successfully to systems where this is not technically true. Thus, noise in the visual pose estimate, presumably stemming from noise at the pixel level and other uncertainties, is approximated as a gaussian based on the average pose error observed above.

Based on the results above, the visual estimator's measurement is expected to vary from the actual position and orientation by $d_m = d_\mu + d_\sigma = 0.1506$ and $\alpha_{max} = \alpha_\mu + \alpha_\sigma = 14.2511°$, respectively. Position error is mapped directly to each axis. To map the orientation error to the expected quaternion perturbation, we assume that α_{max} corresponds to the expected 1-sigma rotation angle of the quaternion orientation, about a random unit axis vector $\mathbf{u}$. Further, since orientation error is assumed to be isotropic (i.e., it can occur around any axis), $\mathbf{u}$ is distributed uniformly around a unit sphere. Finally, we expect that the rotational perturbation has an arbitrarily small, but non-negligible, effect on the scalar component of the quaternion. The covariance model follows as

$$\mathbf{V} = \text{diag}\left(0.1506^2, 0.1506^2, 0.1506^2, 0.2, \frac{1}{3}\cdot\sin^2\left(\frac{14.2511°}{2}\right), \frac{1}{3}\cdot \sin^2\left(\frac{14.2511°}{2}\right), \frac{1}{3}\cdot\sin^2\left(\frac{14.2511°}{2}\right)\right). \quad (28)$$

3.6 IMU calibration

The IMU is calibrated using QGroundControl's [15] built-in calibration helper based on [16] and a simplified calibration model. For accelerometer calibration, the hardware assembly is placed in each of its six principal orientations. Cross-axis relationships are ignored, and the calibration is condensed into six parameters: a scale factor and bias for each of the three axes. For gyroscope and level horizon calibration, only bias is calculated while the assembly is held stationary in its "take-off" orientation on a flat platform (height-adjustable surface with right angled spirit levels). Early trials demonstrated a continuing bias effect, despite the calibration. Hence, the accelerometer and gyroscope biases were adjusted after the initial calibration by their offset from the gravity vector and zero, respectively, calculated as the average from 15000 stationary, level samples:

$$\mathbf{b}_{g_{eff}} = \mathbf{b}_{g_Q} + \mathbf{b}_{g_{stat}} = \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}0.0016\\0.0028\\0.0015\end{bmatrix} \tag{29}$$

$$\mathbf{b}_{a_{eff}} = \mathbf{b}_{a_Q} + \mathbf{b}_{a_{stat}} = \begin{bmatrix}-0.173\\-0.164\\0.329\end{bmatrix} + \begin{bmatrix}-0.1550\\0.0886\\-0.0226\end{bmatrix} \tag{30}$$

The resulting parameters are shown in **Table 2**. The bias and scale factor are applied prior to input into the state estimator, as shown in (31) and (32). Thus, the biases estimated in the 16-element EKF are effectively offsets from these prior biases.

$${}_I\mathbf{u}_a = \mathbf{K}_a * \mathbf{u}_{a_{raw}} - \mathbf{b}_{a_{eff}} \tag{31}$$

$${}_I\mathbf{u}_g = \mathbf{K}_g * \mathbf{u}_{g_{raw}} - \mathbf{b}_{g_{eff}} \tag{32}$$

Table 2. IMU calibration parameters calculated by QGroundControl and stationary correction

Parameter	Accelerometer	Gyroscope
Bias	$\mathbf{b}_{a_Q} = \begin{bmatrix}-0.3280\\-0.0754\\0.3064\end{bmatrix}$ m/s^2	$\mathbf{b}_{g_Q} = \begin{bmatrix}0.0016\\0.0028\\0.0015\end{bmatrix}$ rad/s
Scale factor	$\mathbf{K}_a = \begin{bmatrix}0.992 & 0 & 0\\0 & 0.989 & 0\\0 & 0 & 0.975\end{bmatrix}$	$\mathbf{K}_g = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$
Board static misalignment	${}_B\theta_{x_I} = -0.431°$, ${}_B\theta_{y_I} = -1.874°$ $\therefore {}_B\boldsymbol{R}_I = \begin{bmatrix}0.9995 & -0.0002 & -0.0327\\0.0002 & 1 & 0.0075\\0.0327 & -0.0075 & 0.9995\end{bmatrix}$	

3.7 IMU process noise characterisation

IMU noise was initially characterised based on the Allan variance [17]. However, the resulting Gaussian parameters (squared and multiplied by the expected timestep), when substituted into the process noise covariance matrix of the EKF (**Q**), did not produce accurate results in combination with the noise parameters derived above for the visual state estimator. This is attributed to shortcomings in the visual state estimator characterisation, stemming from the loosely-coupled approach as described in the previous section. Fortunately, the absolute values of the measurement and process noise covariances do not in themselves influence the state estimate from an EKF (although they do affect its certainty); rather, it is the ratio between these that matters. In practice, this means that some trial-and-error is necessary to obtain effective variance parameters in an EKF, particularly if the two sensors have very different noise characteristics. The following covariance matrix was obtained in this manner, and appeared to exhibit good results:

$$\mathbf{Q} = diag[0.01, 0.01, 0.01, 0.2, 0.2, 0.2, 0.01, 0.01, 0.01, 0.001, 0.001, 0.001]. \tag{33}$$

4 Results and discussion

This section presents the results of the evaluations described above, as well as 'ad hoc' data post-processing that was applied to account for phenomena observed in testing and analysis.

4.1 Observations and adjustments

An example of the pose error evaluation result for all trials of a given estimator is shown in Fig. 8. The exponential-type growth observed in the position error in (a) is attributed to image loss or delay. In this case, it is generally a consequence of the wireless transfer of images but is in some cases due to the target going out of frame. In practice, the system is intended to operate without markers and for the camera to have a wired connection to the flight controller; frame losses are therefore expected to be far reduced. Thus, for a more meaningful evaluation, only cases with an interval between corrections of less than one second were considered.

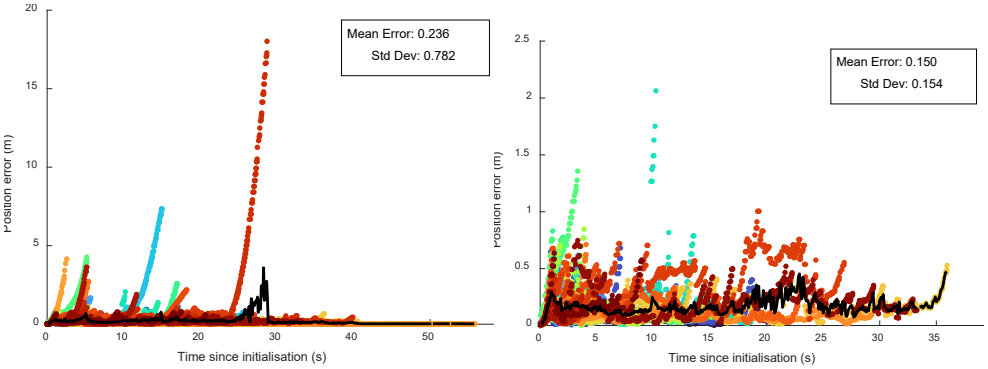

Fig. 8. Position error of sample state estimator for all physical trials: (a) All data, (b) scrubbed of image losses exceeding one second. Different colours represent different trials, black line is the temporal mean.

Further, a consistent orientation error of approximately 7° was observed across all estimators. In addition, in early simulations, it was found that the view from the simulated camera did not match that of the physical camera for the same trajectory. This suggested a misalignment between the motion capture markers and the test platform. A rotation matrix to align these two frames was obtained by using singular value decomposition to solve Wahba's problem for the true and estimated orientation (the latter from the 16-element rectangular EKF for two static trajectories) and then adjusting the result such that the simulated and physical camera images matched. Thus, the approximate misalignment between the motion capture body frame and real body frame (in XYZ rotation order) was obtained as:

$$_{B}\boldsymbol{\theta}_{M} = [8° \quad 1.5° \quad 0.5°]^{\mathrm{T}} \tag{34}$$

This rotation was then used to adjust the existing rotation matrix from the motion capture body to the real body.

4.2 Performance evaluation results

Table 3 shows a summary of the pose errors obtained in physical tests for all the estimators, scrubbed of image losses of over one second.

Table 3. State estimator pose error in physical tests, scrubbed of image losses >1 s

Estimator	Integration method	Orientation error		Position error	
		μ	σ	μ	σ
10-element EKF	Rectangular	2.563	1.628	0.129	0.142
	Trapezoidal	2.563	1.628	0.129	0.142
16-element EKF	Rectangular	3.095	2.048	0.136	0.151
	Trapezoidal	3.147	2.102	0.138	0.152

All state estimators exhibited similar performance, with mean orientation errors varying by less than 5%. Tracking the bias term in the EKF appears to have degraded the pose estimate in all cases. Similarly, trapezoidal integration either made very little difference or degraded the estimate. This may suggest that the additional computational complexity associated with higher order integrations and states is not justified by performance enhancement. However, it is also likely that these estimates have been negatively affected by the slow data rate from the IMU. In practice, state estimators for agile (and inherently unstable) systems such as quadcopters should operate at a rate more in the order of a 100 Hz than 16 Hz. For low update rates, trapezoidal integration causes the effective measurements to be even more out of date than they may already be, thus degrading the state update.

Despite this, the mean position and orientation errors are in line with expectations, as based on the results of the visual state estimator characterisation (Section 1.1) as well as literature [18], which suggests that the estimator successfully disambiguates P3P solutions, and should be sufficient for low-speed applications. Further investigations will be needed to determine the best form of EKF. As it stands, the 10-element versions performed best.

5 Conclusion

This study aimed to develop a computationally lightweight visual–inertial state estimator for practical application and as a learning tool. In terms of practicality, the proposed state estimator demonstrated mean positional and orientation tracking errors of less than 15 cm and 5°, respectively, in physical experiments under non-ideal conditions, without relying on planar constraints. The system therefore shows promise as a lightweight and accurate state estimator and is expected to perform better at higher update rates. However, further evaluations should be conducted to confirm this, and the final system should be designed to be robust to loss of visual markers for a reasonable time period or should ensure that such loss cannot occur. The study is thus effective as a preliminary investigation, and the proposed system achieves its goals of operational simplicity and educational value through its reliance on fundamental algorithms from the computer vision and state estimation fields. Its contributions are therefore a working state estimator, streamlined to run in a single software package, and a clear benchmark for refinement. Future work will involve implementing the state estimator on hardware, testing the fidelity of the simulation environment, and refining the estimation scheme to allow for marker-less pose estimation.

The research reported in this study was partially funded by the Council for Scientific and Industrial Research. The models and used in this study are available on the author's Github repository: github.com/ARamwell/QuadSimEnv/releases/tag/robmech-2025. The data used for the evaluations in this study is available from the authors on request.

References

1. T. Elmokadem, A. V. Savkin, Towards Fully Autonomous UAVs: A Survey, Sensors **21** (2021). 10.3390/s21186223.
2. I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, D. Toal, A survey of state-of-the-art on visual SLAM, Expert Systems with Applications **205**, 117734 (2022). https://doi.org/10.1016/j.eswa.2022.117734.
3. N. Gyagenda, J. V. Hatilima, H. Roth, V. Zhmud, A review of GNSS-independent UAV navigation techniques. Robotics and Autonomous Systems **152**, 104069 (2022). https://doi.org/10.1016/j.robot.2022.104069.
4. M. He, C. Zhu, Q. Huang, B. Ren, J. Liu, A review of monocular visual odometry. Vis. Comput. **36**, 1053–1065(2020). https://doi.org/10.1007/s00371-019-01714-6.
5. E. Olson, AprilTag: A robust and flexible visual fiducial system, in 2011 IEEE Int. Conf. on Robot. and Autom., 3400–3407(2011). https://doi.org/10.1109/ICRA.2011.5979561.
6. L. Kneip, D. Scaramuzza, R. Siegwart, A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation, in CVPR 2011, 2969–2976, Colorado Springs, CO, USA: IEEE, June (2011). https://doi.org/10.1109/CVPR.2011.5995464.
7. L. Kneip, H. Li, Y. Seo, UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability, Computer Vision – ECCV 2014 (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds, Cham: Springer International Publishing, 2014, pp. 127–142). https://doi.org/10.1007/978-3-319-10590-1_9.
8. L. Kneip, M. Chli, R. Siegwart, Robust Real-Time Visual Odometry with a Single Camera and an IMU, in Procedings of the British Machine Vision Conference 2011, 16.1-16.11, Dundee: British Machine Vision Association (2011). https://doi.org/10.5244/C.25.16.
9. S. M. Weiss, Vision based navigation for micro helicopters, Doctoral Thesis, ETH Zurich, 2012. https://doi.org/10.3929/ethz-a-007344020.
10. G. He, Notes on Extended Kalman Filter based Visual-Inertial Navigation Systems, Technical Report (2020). Accessed: Oct. 28, 2024. [Online]. Available: https://www.researchgate.net/publication/344138714_Notes_on_Extended_Kalman_Filter_based_Visual-Inertial_Navigation_Systems
11. L. Kneip, 'MPL@ShanghaiTech'. Accessed: May 11, 2025. [Online]. Available: https://mpl.sist.shanghaitech.edu.cn/Software.html
12. A. Geiger, F. Moosmann, O. Car, B. Schuster, Automatic camera and range sensor calibration using a single shot, in 2012 IEEE Intl. Conf. on Robot. and Autom., 3936–3943, St Paul, MN, USA: IEEE, May (2012). https://doi.org/10.3929/10.1109/ICRA.2012.6224570.
13. 'Holybro Pixhawk 6C | PX4 Guide (main)'. Accessed: May 25, 2025. [Online]. Available: https://docs.px4.io/main/en/flight_controller/pixhawk6c.html
14. 'Prime^{x} 13 - In Depth', OptiTrack. Accessed: May 26, 2025. [Online]. Available: http://optitrack.com/cameras/primex-13/index.html
15. 'Sensor Setup (PX4) | QGC Guide (master)'. Accessed: May 26, 2025. [Online]. Available: https://docs.qgroundcontrol.com/master/en/qgc-user-guide/setup_view/sensors_px4.html

16. D. Tedaldi, A. Pretto, E. Menegatti, A robust and easy to implement method for IMU calibration without external equipments, in 2014 IEEE Intl. Conf. on Robot. and Autom. (ICRA), 3042–3049, May (2014) pp. 3042–3049. https://doi.org/10.3929/10.1109/ICRA.2014.6907297.
17. M. B. Marinov, B. Ganev, N. Djermanova, T. D. Tashev, Analysis of Sensors Noise Performance Using Allan Deviation, in *2*019 IEEE XXVIII International Scientific Conference Electronics (ET), 1-4, Sept. (2019). https://doi.org/10.3929/10.1109/ET.2019.8878552.
18. A. Breitenmoser, L. Kneip, R. Siegwart, A monocular vision-based system for 6D relative robot localization, in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 79-85, Sept. (2011). https://doi.org/10.3929/10.1109/IROS.2011.6094851.