Master of Science in Informatics at Grenoble
Master Mathématiques Informatique - spécialité Informatique
option Graphics, Vision, Robotics

Absolute scale velocity determination combining visual and inertial measurements for micro aerial vehicles

Jacques Kaiser

24th of June 2015

Research project performed at Inria Grenoble Rhône-Alpes

Under the supervision of:
Dr Agostino Martinelli

Defended in front of a jury composed of:
Prof James Crowley
Dr Edmond Boyer
Dr Dominique Vaufreydaz
Dr Jean-Sébastien Franco
Dr Guillaume Huard
Dr Frédéric Devernay

June 2015

Abstract

State of the art approaches for visual-inertial sensor fusion are filter based algorithms. These methods are recursive by design and therefore require an initialization. Due to the nonlinearity of these systems, a poor initialization can have a dramatic impact on the performance of the estimations. On the other hand, methods that work without initialization have been developed in computer vision but they can only determine physical quantities up to a scale.

In this report we evaluate a novel closed-form solution that recovers the initial velocity, the attitude and the absolute scale by fusing visual and inertial measurements. By nature, a closed-form solution is deterministic and thus does not require any initialization.

By experimenting the closed-form solution with real and synthetic data, we were able to improve its robustness against noisy measurements. The main contribution of this report is the introduction of a simple method that automatically estimates the gyroscope bias which was a major performance bottleneck. Compared to the original method, the new method is now robust to this bias, and also provides the gyroscope bias.

Résumé

La fusion de données visuo-inertielles se fait classiquement avec des algorithmes de filtrage. Ces méthodes sont récursives par nature et ainsi nécéssitent une initialisation externe. Une mauvaise initialisation peut avoir des conséquences dramatiques sur les performances des estimations, dù à la non-linéarités de ces systèmes. D'autre part, des méthodes déterministes qui ne tiennent compte que des mesures visuelles ont été introduite dans le cradre de la vision par ordinateur. En revanche, ces méthodes ne peuvent estimer les quantitées physiques qu'à un facteur d'échelle près.

Dans ce rapport, nous évaluons un récente solution analytique qui exprime la vitesse initiale, l'orientation et le facteur d'échelle en fonction des mesures visuelles et inertielles. Cette solution est déterministe et ne requiert pas d'initialisation.

En évaluant cette méthode sur des données terrain et synthétisées, nous avons amélioré sa robustesse contre des mesures érronées. La contribution principale de ce rapport est l'introduction d'une méthode simple pour le calcul automatique de l'erreur systématique du gyroscope qui endommage la qualité des évaluations. Comparé à la méthode originale, notre nouvelle méthode est désormais robuste à cette erreur systématique et calcul cette erreur.

Acknowledgement

This work was supported by the French National Research Agency ANR 2014 through the project VIMAD

Contents

— 1 —
Introduction

Autonomous mobile robots navigating in unknow environments have an intrinsic need to perform localization and mapping using only on-board sensors. Concerning Micro Aerial Vehicles (MAV), a critical issue is to limit the number of on-board sensors to reduce weight and power consumption. Therefore, a common setup is to combine a monocular camera with an inertial measurements unit (IMU). On top of being cheap, these sensors have very interesting complementary. Adadditionally, they can operate in indoor environments where Global Positioning System (GPS) signals are shadowed. An open question is how to optimally fuse the information provided by these sensors.

This question has been well studied in the framework of robotics, computer vision and neuroscience. We can distinguish two sub-problems: Visual-Inertial Structure from Motion (Vi-SfM) and Visual Inertial Odometry (VIO).

Both Vi-SfM and VIO fuse the same information but aim at recovering different state. Specifically, Vi-SfM estimates three-dimensional structure of the environment while VIO determines the position and orientation of the robot. Thus Vi-SfM relates to mapping while VIO relates to localization. Simultaneous Localization and Mapping (SLAM) consists of both recovering a map of the environment and localizing the robot within it.

Currently, most sensor fusion algorithms are either **filter based or iterative**. That is, given a current state and measurements, they return an updated state. While working well in practice, these algorithms need to be **provided by an external initial state**.

If we assume that the MAV will take off from a static standing position on the floor, we can initialize its speed and attitude to zeros. Moreover, we could initialize the global scale factor using a stereo pair, depth camera or defocus. In general, an initialization of the global scale factor is obtained from a known metric in the environment or by a cumbersome pre-calibration routine. However, such initialization methods would not work to recover the state after a system failure, such as a kidnapping. It would not work either for more dynamic take off, such as parachuting a MAV from a plane.

The initialization of a filter based method is critical. Due to non-linearity of the system, a poor initialization can result into converging towards local minima and providing faulty states with high confidence. Indeed, another shortcoming of filters is that they can silently fail.

In this report we demonstrate the efficiency of a **recent closed-form solution** that fuses visual and inertial data to obtain the structure of the environment at the global scale along with the attitude and the speed of the robot. That is, solving both the Vi-SfM and VIO problems.

By nature, a closed-form solution is deterministic and thus **does not require any initialization**. It is assumed that the camera is calibrated and the transformation between the IMU and

the camera is known. This is a fair assumption for industrial drones to come pre-calibrated.

In this report, we will start by introducing the closed-form solution from an intuitive, mathematical point of view as it was introduced in [18].

In chapter 4 we will discuss implementation issues when running the closed-form solution on real data. Specifically, we will modify the original expression of the closed-form solution in order to reduce numerical errors.

Moreover, the impact of biased inertial sensors will be studied. Despite the case of a biased accelerometer was considered in [18], we found by experiment that its influence on the performance of the closed-form solution is negligible. On the other hand, we realized that the performance of the method is **strongly affected by the gyroscope bias**. We then introduce a simple method that automatically estimates this bias using the closed-form solution. By adding this new method for the bias estimation to the original method we obtain results which are equivalent to the ones in absence of bias. Compared to the original method, the new method is now robust to the gyroscope bias, and also **provides the gyroscope bias**. We also present a tweak of this method that increases its robustness when only a small amount of measurements is available.

The contribution of this report are:

- Reformulation of the original closed-form solution for numerical stability in Section 4.2;
- Extensive experiments with real and synthetic data to evaluate the impact of biased inertial measurements on the performance in Section 4.3;
- Simple method for online computation of the gyroscope bias using the closed-form solution in Section 5.2;
- Adaptation of this method for a low amount of available measurements in Section 5.3.2.

We validate our method by comparing it by a state of the art Kalman Filter performing visual-inertial sensor fusion as described in [26].

— 2 —
Related work

The problem of fusing vision and inertial data has been extensively investigated in the past. A special issue of the International Journal of Robotics Research has recently been devoted to this important topic [4]. The majority of the approaches so far introduced perform the fusion of vision and inertial sensors by filter-based algorithms. These algorithms are recursive: they provide an updated state from an old state given recent visual and inertial measurements. The most popular approaches to sensor fusion are extensions of the Kalman Filter.

The Kalman filter is statistically optimal for linear system but by design does not work with nonlinear systems. Both Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) allow the use of nonlinear functions at the cost of the optimality.

The success of these methods relies on a proper initialization of the state vector. Indeed, because of the system non-linearities, **lack of precise initialization can irreparably damage the entire estimation process**. However, in literature, this initialization is often guessed or assumed to be known [2][15][11][3].

We are therefore interested into a **deterministic solution** that analytically expresses all the state in terms of the measurements provided by the sensors during a short time-interval.

Some deterministic solutions have been introduced in the field of computer vision and rely only on visual measurements. These techniques have two major drawbacks. Firstly, it is theoretically **impossible to recover the absolute scale**. That is, any estimation (speed of the platform, distance to features) can be determined only up to scale. Obviously, the knowledge of this absolute scale is critical for a MAV performing autonomous navigation tasks. Secondly, since the camera does not feel the gravity, it is also theoretically **impossible to recover the absolute orientation** of the platform. Again, when considering the case of an autonomous MAV this knowledge is critical too. At least the absolute roll and pitch angles are essential in order to maintain the MAV in a stable hover position.

2.1 Deterministic solution from visual measurements

The 8-point algorithm introduced by [16] in 1981 is a milestone in Computer Vision. It estimates the essential matrix related to a stereo camera pair from a set of corresponding image points.

This algorithm has been improved in [10] to give an accurate estimation of the fundamental matrix, thus allowing the use of uncalibrated camera. It has also been proven that only five

points can be used but the system of equations to be solved becomes more complicated [21]. A simpler implementation is proposed in [14] .

From the essential matrix, it is possible to recover the relative rotation and translation between the two camera poses, but only up to scale [9]. Moreover, visual measurements alone do not provide the absolute roll and pitch angles. In [8] they initialize the camera pose using a known landmark. They can therefore recover the absolute roll and pitch angles of the platform, however this method is not suited to unknown environments.

In order to overcome these limitations it is required to take into consideration the inertial measurements.

2.2 Deterministic solution from visual and inertial measurements

Inertial measurements have been used along with visual measurements to improve over the aforementioned methods. A 4-point algorithm more accurate than its 5-point counterpart assuming a known relative rotation angle is proposed in [13]. Their approach has the advantage that the IMU camera calibration is not required. However the absolute scale, roll and pitch angles are not recovered.

In [17] the authors pre-integrate their IMU data in an arbitrary frame moving at the same velocity than the previous body frame of the MAV. Since the reference frame is moving at the same instantaneous velocity as the vehicle was at the previous pose, the initial velocity with respect to this frame is zero, making integration without initial conditions possible. The camera measurements are then used to match the initial reference frame with the defined reference frame. Their method can recover the absolute scale, roll and pitch angles, in a linear manner. However, their method is purely numerical and we can not analytically derive its properties.

A closed-form solution is provided in [5] to determine the absolute scale. However they assume an additional on-board sensor measuring a metric quantity, such as an altimeter or an air pressure sensor. It is therefore not applicable for our minimal configuration.

A procedure to quickly re-initialize MAV after a failure is discussed in [6]. However, their method requires an altimeter to estimate the absolute scale.

Recently, a closed-form solution has been introduced in [19]. From integrating inertial and visual measurements over a short time-interval, this solution provides the absolute scale, roll and pitch angles, initial velocity and distance to features. Specifically, all the physical quantities are obtained by simply **inverting a linear system**. The solution of the linear system can be refined with a quadratic equation assuming the knowledge of the gravity magnitude.

This closed-form has been improved in [15] such that it also estimates the translation between the camera and the IMU. The author also proposes a way to compute the relative rotation between the camera and the IMU. Their method is therefore independent of external IMU camera calibration, hence well suited for a power-on-and-go system.

A more intuitive expression of the same closed-form solution is derived in [18]. This formulation also provides the accelerometer bias, which is expressed as an unknown in the linear system.

In this report we carry out an analysis of this closed-form solution to find out its limitations. Specifically, we show that it is resilient to the accelerometer bias but strongly affected by the gyroscope bias. We then introduce a simple method that automatically estimates this bias. By adding this new method for the bias estimation to the original method we obtain results which

are equivalent to the ones in absence of bias. Compared to the original method, the new method is now robust to the gyroscope bias, and also provides the gyroscope bias.

— 3 —
Derivation of the closed-form solutions

3.1 The considered system

We consider a system (from now on we call it the platform) consisting of a monocular camera and an inertial measurement unit (IMU). The IMU consists of a 3-axis accelerometer and a 3-axis gyroscope.

Note that in practice, the platform is a rigid body, on which sits the camera and the IMU at different position. In order to simplify our model, we coincide the camera frame with the IMU frame. Indeed, assuming that the transformations among the IMU frame and the camera frame are known, we can transform measurements from one frame to another. More details on this are given in appendix 7.

We refer to the coincided frame by **local frame**. Its origin is the position of the platform. We will adopt **upper-case letters** to denote vectors in this frame.

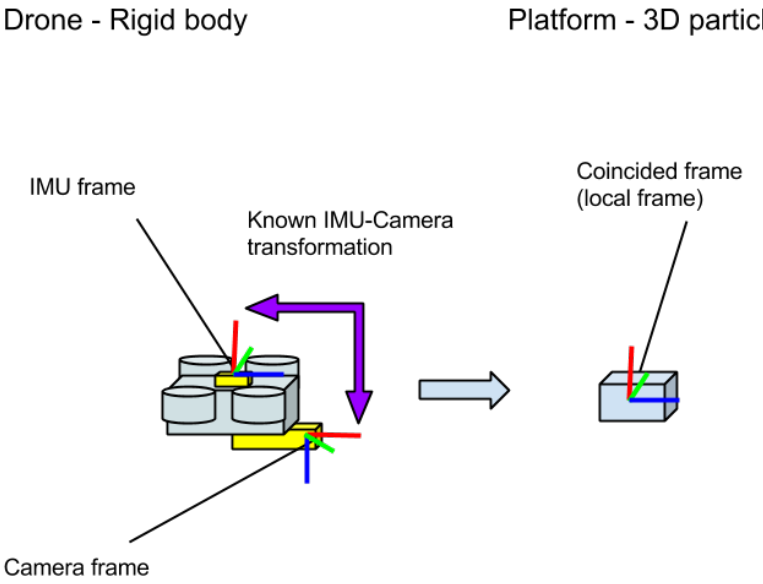


Figure 3.1 – Local frame

Since the platform is moving in a 3D environment, we introduce a **global frame** in order to characterize its motion. Its z-axis points vertically upwards. We will adopt **lower-case letters**

to denote vectors in the global frame (e.g. the gravity is $g = [0,0,-\mathrm{g}]^T$, where $\mathrm{g} \simeq 9.8ms^{-2}$).

Since the local frame is in motion, we adopt the following notation: $V_t(\tau)$ will be the vector with global coordinates $v(\tau)$ in the local frame at time t. Additionally, we will denote with $C_{t_1}^{t_2}$ the matrix which characterizes the rotation that occurred during the time interval (t_1, t_2) and with $C_{t_2}^{t_1}$ its inverse (i.e., $(C_{t_1}^{t_2})^{-1} = (C_{t_1}^{t_2})^T = C_{t_2}^{t_1}$).

Let us refer to vectors which are independent of the origin of the reference frame (e.g., speed, acceleration, etc.). To transform these vectors from one frame to another we just have to consider the rotation that occurred between these two frames, regardless of any translation. For these vectors we have: $V_{t_1}(\tau) = C_{t_1}^{t_2} V_{t_2}(\tau)$. Finally, C^t will denote the rotation matrix between the global frame and the local frame at time t, i.e. $v(\tau) = C^t V_t(\tau)$.

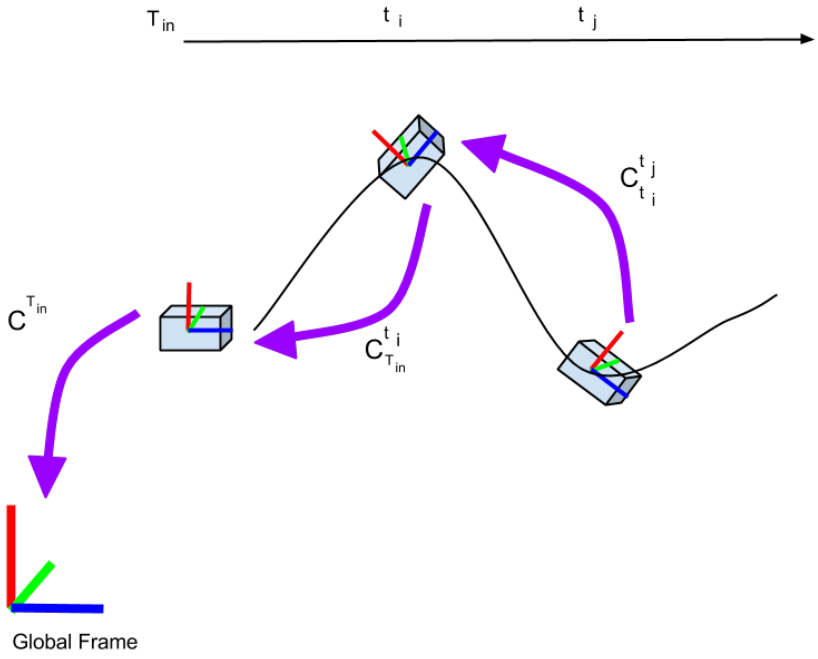


Figure 3.2 – Frame transformations

The IMU provides the platform angular speed and linear acceleration. Regarding the acceleration, the one perceived by the accelerometer (A) is not simply the inertial acceleration ($A^{inertial}$). It also includes the gravitational acceleration (G).

We assume that the camera is observing one or more point features during the time interval $[T_{in}, T_{fin}]$. Moreover, we assume the intrinsic camera parameters to be known. That is, given a 3D point p^i depicting a point feature that we observe at time t, we can compute the direction towards this point feature from the local frame at time t. In other words, we can compute the vector $P_t^i(t)$ up to scale. More details on this are given in appendix 7.

3.2 Differential equation for generic 3D-motion

Our goal is to express in closed-form the platform velocity, roll and pitch angles and the absolute scale at a given time T_{in} only in terms of the visual and inertial measurements obtained during the time interval $[T_{in}, T_{fin}]$.

Let us introduce the following notation:

- r is the position of the platform at a given time $t \in [T_{in}, T_{fin}]$;
- $v = \dot{r}$ is the speed of the platform at a given time $t \in [T_{in}, T_{fin}]$;
- $a = \ddot{r}$ is the acceleration of the platform at a given time $t \in [T_{in}, T_{fin}]$.

We can express the position of the platform at time t with respect to its acceleration a, the initial speed $v(T_{in})$ and the initial position $r(T_{in})$:

$$\begin{aligned} \ddot{r}(t) &= a(t) \\ \dot{r}(t) &= v(T_{in}) + \int_{T_{in}}^{t} a(\tau)\mathrm{d}\tau \\ r(t) &= r(T_{in}) + v(T_{in})\Delta_t + \int_{T_{in}}^{\tau}\int_{T_{in}}^{t} a(\xi)\mathrm{d}\xi\,\mathrm{d}\tau \end{aligned}$$

With $\Delta_t = (t - T_{in})$. The last term contains a double integral over time, which can be simplified in a single integral by integrating by parts. We obtain:

$$r(t) = r(T_{in}) + v(T_{in})\Delta_t + \int_{T_{in}}^{t} (t-\tau)a(\tau)\mathrm{d}\tau \tag{3.1}$$

This is a well-known equation in physics, expressing the position of a particle in motion. Our goal is to link this equation with the measurements provided by our sensors, expressed in the local frame.

3.3 Inertial Measurements

The accelerometer does not provide the vector $a(\tau)$. It provides the acceleration in the local frame and it also perceives the gravitational component. In other words, the accelerometer provides the vector:

$$A_\tau(\tau) = A_\tau^{inertial}(\tau) - G_\tau \tag{3.2}$$

Note that the gravity comes with a minus since, when the platform does not accelerate, the accelerometer perceives an acceleration which is the same of an object accelerated upward in absence of gravity.

Note also that the vector G_τ, the gravity expressed in the local frame, depends on time only because the local frame can rotate.

We want to highlight the vector $A_\tau(\tau)$ provided by the accelerometer in the equation of the position of the platform 3.1. Let us express the global inertial acceleration into the local reference frame:

$$a(\tau) = C^{T_{in}} C_{T_{in}}^{\tau} A_\tau^{inertial}(\tau)$$

By injecting 3.2 we obtain an expression of the global inertial acceleration with respect to the accelerometer measurements:

$$\begin{aligned} a(\tau) &= C^{T_{in}} C^{\tau}_{T_{in}} \left(A_\tau(\tau) + G_\tau\right) \\ &= C^{T_{in}} C^{\tau}_{T_{in}} A_\tau(\tau) + C^{T_{in}} C^{\tau}_{T_{in}} G_\tau \\ &= C^{T_{in}} C^{\tau}_{T_{in}} A_\tau(\tau) + g \end{aligned} \tag{3.3}$$

We can now inject 3.3 back into the equation of the position of the platform 3.1:

$$\begin{aligned} r(t) &= r(T_{in}) + v(T_{in})\Delta_t + \int_{T_{in}}^{t} (t-\tau)\left(C^{T_{in}} C^{\tau}_{T_{in}} A_\tau(\tau) + g\right) \mathrm{d}\tau \\ &= r(T_{in}) + v(T_{in})\Delta_t + \int_{T_{in}}^{t} \left[(t-\tau)\left(C^{T_{in}} C^{\tau}_{T_{in}} A_\tau(\tau)\right) + g(t-\tau)\right] \mathrm{d}\tau \\ &= r(T_{in}) + v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + \int_{T_{in}}^{t} (t-\tau) C^{T_{in}} C^{\tau}_{T_{in}} A_\tau(\tau) \mathrm{d}\tau \\ &= r(T_{in}) + v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + C^{T_{in}} \int_{T_{in}}^{t} (t-\tau) C^{\tau}_{T_{in}} A_\tau(\tau) \mathrm{d}\tau \\ &= r(T_{in}) + v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + C^{T_{in}} S_{T_{in}}(t) \end{aligned} \tag{3.4}$$

where:

$$S_{T_{in}}(t) = \int_{T_{in}}^{t} (t-\tau) C^{\tau}_{T_{in}} A_\tau(\tau) \mathrm{d}\tau \tag{3.5}$$

Observe that $A_\tau(\tau)$ is directly provided by the accelerometer. Moreover, $C^{\tau}_{T_{in}}$ can be obtained by integrating the gyroscope measurements [7]. Therefore, $S_{T_{in}}(t)$ is fully determined by the IMU. The unknowns in this equations are $r(T_{in})$, $v(T_{in})$ and $C^{T_{in}}$, respectively the initial position, the initial velocity and the transformation from the global to the initial local reference frame. We will now add more constraints by taking into account the camera observations.

3.4 Visual Measurements

We use our camera as a bearing sensor providing the direction towards point-features in the local frame. The method we use to extract point-features from an image and match them from an image to another are explained in appendix 7. This requires the camera to be calibrated.

Let us suppose that N point-features are observed, simultaneously. Let us denote their position in the physical world with $p^i, i = 1, ..., N$. According to our notation, $P^i_t(t)$ will denote their position at time t in the local frame at time t. Note that, unlike the acceleration, these vectors are not independent of the origin of the reference frame. We have:

$$p^i = r(t) + C^{T_{in}} C^{t}_{T_{in}} P^i_t(t) \tag{3.6}$$

Moreover, for the first observation at time $t = T_{in}$ we have

$$\begin{aligned} & p^i &= r(T_{in}) + C^{T_{in}} P^i_{T_{in}}(T_{in}) \\ \Leftrightarrow\ & p^i - r(T_{in}) &= C^{T_{in}} P^i_{T_{in}}(T_{in}) \end{aligned} \tag{3.7}$$

We can inject the equation of the position of the platform 3.4 into the equation of a bearing 3.6 and simplifying with the equation of the initial bearing 3.7:

$$
\begin{aligned}
& p^i && = r(T_{in}) + v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + C^{T_{in}} S_{T_{in}}(t) + C^{T_{in}} C^t_{T_{in}} P^i_t(t) \\
\Leftrightarrow\ & p^i - r(T_{in}) && = v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + C^{T_{in}} S_{T_{in}}(t) + C^{T_{in}} C^t_{T_{in}} P^i_t(t) \\
\Leftrightarrow\ & C^{T_{in}} P^i_{T_{in}}(T_{in}) && = v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + C^{T_{in}} S_{T_{in}}(t) + C^{T_{in}} C^t_{T_{in}} P^i_t(t)
\end{aligned}
$$

Now we want to express this equation in the local reference frame. We remind the reader that, according to our notation, $v(T_{in}) = C^{T_{in}} V_{T_{in}}(T_{in})$ and $g = C^{T_{in}} G_{T_{in}}$.

We can therefore further simplify our equation by multiplying by $(C^{T_{in}})^{-1}$, the matrix that transforms vectors from the global to the local reference frame at time T_{in}:

$$
\begin{aligned}
P^i_{T_{in}}(T_{in}) &= (C^{T_{in}})^{-1} v(T_{in})\Delta_t + (C^{T_{in}})^{-1} g\frac{\Delta_t^2}{2} + S_{T_{in}}(t) + C^t_{T_{in}} P^i_t(t) \\
&= V_{T_{in}}(T_{in})\Delta_t + G_{T_{in}}\frac{\Delta_t^2}{2} + S_{T_{in}}(t) + C^t_{T_{in}} P^i_t(t) \qquad (3.8) \\
\Leftrightarrow\ S_{T_{in}}(t) &= P^i_{T_{in}}(T_{in}) - V_{T_{in}}(T_{in})\Delta_t - G_{T_{in}}\frac{\Delta_t^2}{2} - C^t_{T_{in}} P^i_t(t)
\end{aligned}
$$

A single image provides the bearing angles of all the point-features in the local frame. In other words, an image taken at time t provides all the vectors $P^i_t(t)$ up to scale for $i = 1, ..., N$.

We assume that the camera provides n_i images of the same N point-features taken at time t_i such that $T_{in} = t_1 < t_2 < ... < t_{n_i} = T_{fin}$. Moreover, without loss of generality, we can set $T_{in} = 0$, i.e. $\Delta_t = t$.

From now on, for the sake of simplicity, we adopt the following notation:

- $P^i_j \equiv C^{t_j}_{T_{in}} P^i_{t_j}(t_j), i = 1, 2, ..., N; j = 1, ..., n_i$
- $V \equiv V_{T_{in}}(T_{in})$
- $G \equiv G_{T_{in}}$
- $S_j \equiv S_{T_{in}}(t_j), j = 1, ..., n_i$

With this notation, the P^i_j vectors are expressed in the local frame at time T_{in}. This is equivalent to rotating all local frames at time t to the initial local frame at time T_{in}. Note that we can determine them up to scale since we can determine the rotation matrix $C^{t_j}_{T_{in}}$ with the gyroscope.

Let us denote μ^i_j the unit vector with the same direction of P^i_j. For all point-features we introduce the scale factor λ^i_j such that $P^i_j = \lambda^i_j \mu^i_j$.

We can rewrite 3.8 with our new notation:

$$
S_j = \lambda^i_1 \mu^i_1 - V t_j - G\frac{t_j^2}{2} - \lambda^i_j \mu^i_j \qquad (3.9)
$$

We will now highlight the underlying linear system.

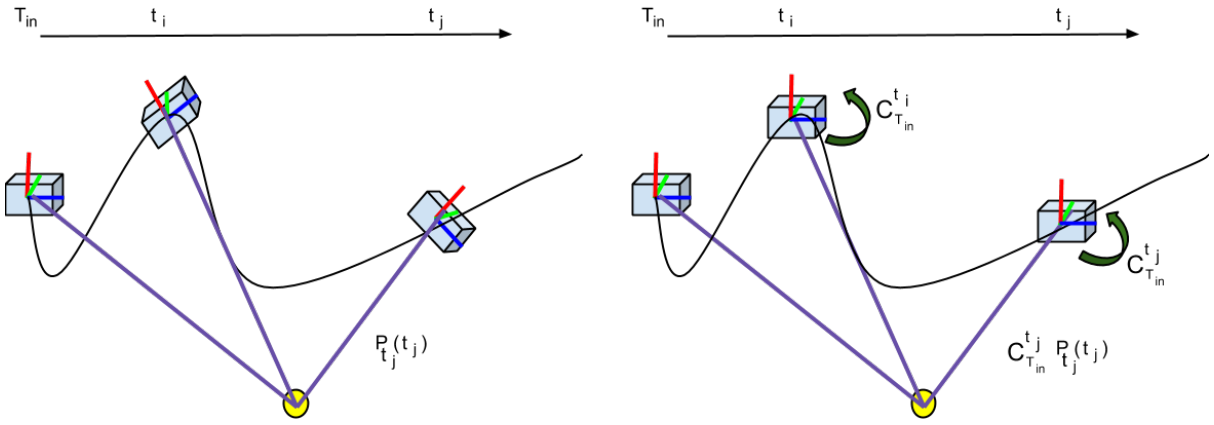


Figure 3.3 – Express bearings in local frame at time T_{in}

3.5 Linear System

The VIO is the determination of V, the velocity in the local frame, along with the orientation. Note that determining G, the gravity vector in the local frame, is equivalent to determining the absolute roll and pitch angles of the platform. These vectors being three dimensional, they together yield six unknowns.

The Vi-SfM problem is the determination of the position of the N point-features in the local frame. In our case this is equivalent to determining the scale factors λ_1^i for $i = 1,...,N$. The use of 3.9 requires us to also determine the quantities λ_j^i, the distance to the point-features at each observation. This yields a total of $n_i * N$ unknowns.

As Equation 3.9 holds for each three dimensions of all point-features $i = 1,...,N$ and each observation starting from the second one $j = 2,...n_i$, we therefore have a system consisting of $3(n_i - 1)N$ equations in $6 + Nn_i$ unknowns. Indeed, note that when the first observation occurs, at $t_j = T_{in} = 0$, the Equation 3.9 is always satisfied and therefore does not provide information. We can therefore write our system using matrix notations. Solving the system is equivalent to inverting a matrix of $3(n_i - 1)N$ rows and $6 + Nn_i$ columns.

In [18], the author proceeded to one more step before expressing the underlying linear system. For an observation at time t_j, they subtract the equation of the first point-feature $i = 1$ happening at time j to all other point-features $1 < i <= N$ at time j. The system therefore becomes:

$$\left[\begin{array}{lcl} S_j & = & \lambda_1^i \mu_1^i - V t_j - G\frac{t_j^2}{2} - \lambda_j^i \mu_j^i \\ 0_3 & = & \lambda_1^1 \mu_1^1 - \lambda_j^1 \mu_j^1 - \lambda_1^i \mu_1^i + \lambda_j^i \mu_j^i \end{array}\right. \tag{3.10}$$

We will write 3.10 as a linear system. Let us define the two column vectors X and S:

$$\begin{array}{rcl} X & \equiv & [G^T, V^T, \lambda_1^1, ..., \lambda_1^N, ..., \lambda_{n_i}^1, ..., \lambda_{n_i}^N]^T \\ S & \equiv & [S_2^T, 0_3, ...0_3, S_3^T, 0_3, ..., 0_3, ..., S_{n_i}^T, 0_3, ..., 0_3]^T \end{array}$$

and the matrix:

$$\Xi \equiv \left[\begin{array}{c|c|c|c|c|c|c|c|c|c|c} T_2 & S_2 & \mu_1^1 & 0_3 & 0_3 & -\mu_2^1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_{33} & 0_{33} & \mu_1^1 & -\mu_1^2 & 0_3 & -\mu_2^1 & \mu_2^2 & 0_3 & 0_3 & 0_3 & 0_3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0_{33} & 0_{33} & \mu_1^1 & 0_3 & -\mu_1^N & -\mu_2^1 & 0_3 & \mu_2^N & 0_3 & 0_3 & 0_3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ T_{n_i} & S_{n_i} & \mu_1^1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\mu_{n_i}^1 & 0_3 & 0_3 \\ 0_{33} & 0_{33} & \mu_1^1 & -\mu_1^2 & 0_3 & 0_3 & 0_3 & 0_3 & -\mu_{n_i}^1 & \mu_{n_i}^2 & 0_3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0_{33} & 0_{33} & \mu_1^1 & 0_3 & -\mu_1^N & 0_3 & 0_3 & 0_3 & -\mu_{n_i}^1 & 0_3 & \mu_{n_i}^N \end{array}\right]$$

Where $T_j \equiv \frac{t_j^2}{2}$, $S_j \equiv -t_j I_3$ and I_3 is the identity 3 x 3 matrix; 0_{33} is the 3 x 3 zero matrix. The linear system in Equation 3.10 can be written in the following compact format:

$$\Xi X = S \tag{3.11}$$

The sensor information is completely contained in the above linear system. Additionally, in [18], the author added a quadratic equation assuming the gravitational acceleration is a priori known. Let us denote the gravitational magnitude by g. We have the extra constraint $|G| = g$. We can express this constraint in matrix formulation:

$$|\Pi X|^2 = g^2 \tag{3.12}$$

With $\Pi \equiv [I_3, 0_3, ..., 0_3]$. The VIO and the Vi-SfM problems can be solved by finding the vector X which satisfies 3.11 and 3.12.

In the next chapters, we will evaluate the performance of this method on real terrain data. This will allow us to identify its weaknesses and bring modifications to overcome them.

— 4 —

Implementation of the closed-form solutions

The method we use to solve the overconstrained linear system $\Xi X = S$ is a singular value decomposition (SVD) since it yields numerically robust solutions.

In the following plots, the IMU data we used were obtained from terrain acquisitions, while we simulated the point feature observations. This separation allowed us to better understand the weaknesses of our method. Moreover, the MAV was tracked with an optical Vicon tracking system allowing us to compare our estimations with the ground truth.

We use single value measure of error to evaluate the performance of our method. Specifically, for vectors such as speed and gravity, we compute the error with an euclidean distance of the estimated vector and the ground truth, normalized by the ground truth. We measure our error on the absolute scale by computing the mean error over all estimated distances to point features λ_j^i

In general, we use one camera observation every 0.3 seconds, even if the camera provides significantly more frames. Indeed, we can discard most of the camera observations. If two observations are too close to each other, then the additional equations do not bring much information to our system. Reducing the number of considered frames reduces the size of the matrices, thus speeds up the computations.

As an example, over a time interval of 3 seconds, we obtain 11 distinct frames. When observing 7 features, it yields a system of $3 \times 10 \times 7 = 210$ equations and $6 + 7 \times 10 = 76$ unknowns.

In this chapter, we will start by presenting the results obtained with the original closed-form solution on terrain data. Our goal is to identify its performance bottlenecks and introduce modifications to overcome them.

The case of a biased gyroscope is addressed in chapter 5. Indeed, the main contribution of this report is the simple method introduced in Section 5.2 to compute the gyroscope bias with the closed-form solution, which happened to be a major performance bottleneck.

4.1 Original closed-form solution

The original closed-form solution described in [18] will be used as a basis for our work. Moreover, we can also use the knowledge of the gravity magnitude in order to refine our results with Equation 3.12. In this case, we are minimizing a linear objective function with a quadratic constraint. in Figure 4.1, we represent the quality of the evaluations with and without this additional constraint.

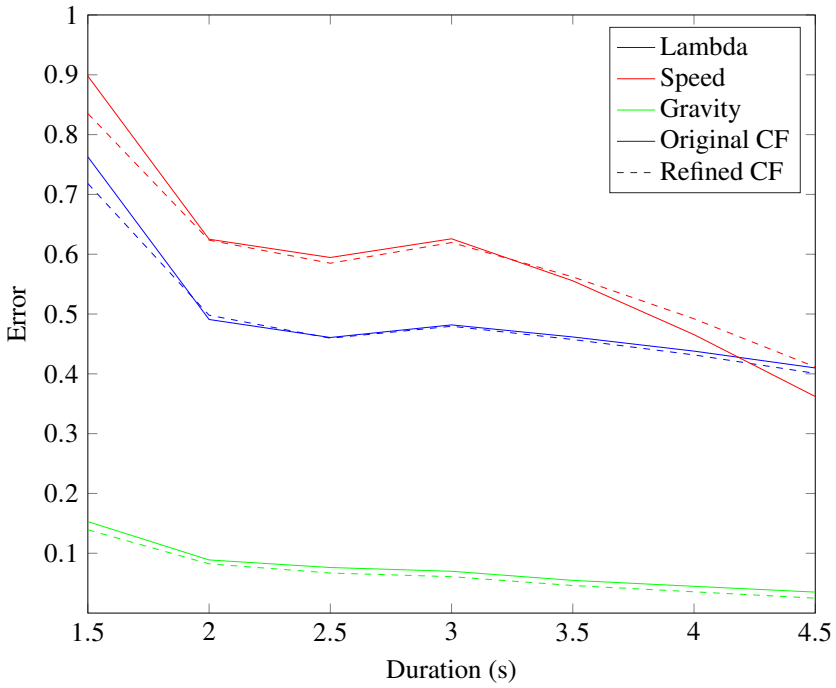


Figure 4.1 – Original closed-form solution estimations with and without gravity knowledge refinement. We are observing 7 features over a variable duration of integration.

Note how the evaluations get better as we increase the integration duration. Indeed, our equations come from an extended triangulation [19]. Therefore, it requires a significant difference in the measurements over time to robustly estimate the state. Also note that the gravity is robustly estimated, whereas the estimations of the speed and the distance to the features is more erroneous.

Since the gravity is well estimated with the standard closed-form, it comes without surprise that constraining its magnitude does not impact the results much. That is why the estimations with and without the refinement are so similar.

With around 45% of error on the speed and the distance to the features after 4 seconds of integration, the original closed-form solution does not perform very well on terrain data. In the following sections, we will introduce modifications to improve over these results.

4.2 Numerical stability

The additional step of subtracting equations with the first point-feature bearing in [18] was taken for observability analysis purposes. However, when purely solving the closed-form solution, we have found that not including this additional step yielded better results.

In other words, we compared the two formulations of the linear system:

- the formulation in 3.11 proposed by [18];
- the formulation directly obtained from 3.9 without the additional step of subtracting the equation of the first point-feature to all other point-features.

In other words, for the second formulation, X remains unchanged but the vector S and the matrix Ξ become:

$$S \equiv [S_2^T, ..., S_2^T, S_3^T, ..., S_3^T, ..., S_{n_i}^T, ..., S_{n_i}^T]^T$$

$$\Xi \equiv \left[\begin{array}{c|c|c|c|c|c|c|c|c|c|c} T_2 & S_2 & \mu_1^1 & 0_3 & 0_3 & -\mu_2^1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ T_2 & S_2 & 0_3 & \mu_1^2 & 0_3 & 0_3 & -\mu_2^2 & 0_3 & 0_3 & 0_3 & 0_3 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ T_2 & S_2 & 0_3 & 0_3 & \mu_1^N & 0_3 & 0_3 & -\mu_2^N & 0_3 & 0_3 & 0_3 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ T_{n_i} & S_{n_i} & \mu_1^1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\mu_{n_i}^1 & 0_3 & 0_3 \\ T_{n_i} & S_{n_i} & 0_3 & \mu_1^2 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\mu_{n_i}^2 & 0_3 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ T_{n_i} & S_{n_i} & 0_3 & 0_3 & \mu_1^N & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\mu_{n_i}^N \end{array}\right]$$

In our experiments, the second formulation always obtained better results than the original one, especially in the presence of many features.

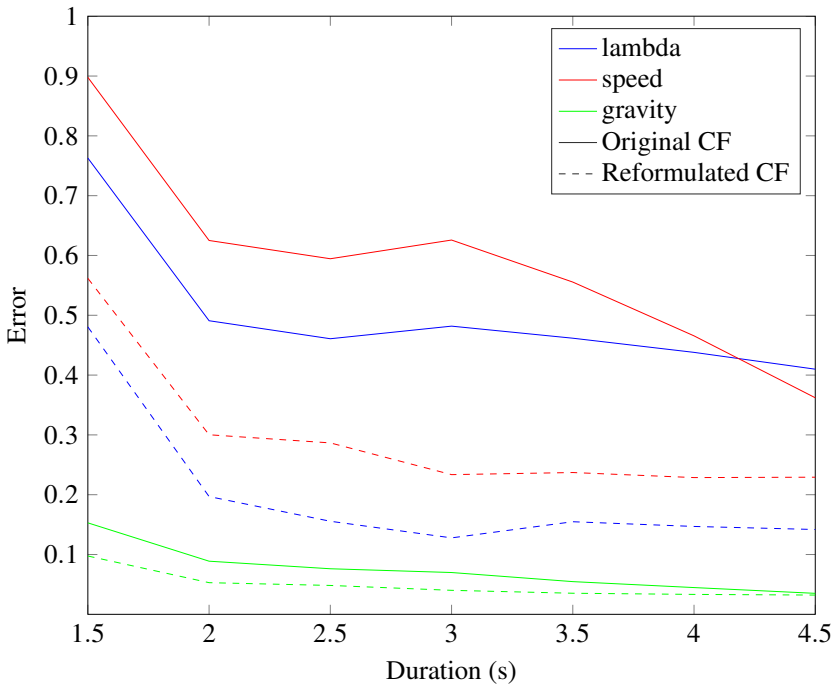


Figure 4.2 – Estimation error of the original formulation of the closed-form solution against its reformulation. We are observing 7 features over a variable duration of integration.

The difference in accuracy is considerable, see Figure 4.2. In the original formulation, because we subtract the equations concerning the first point-feature to all other equations, we grant more weight to the bearings of the first point-feature than to all other point-features. Indeed, any error in the bearing of the first point-feature will be propagated to all other equations.

This is equivalent to estimating the mean of a probability distribution function by subtracting the first measurement to all others. Let $N(\mu, \sigma)$ be a normal distribution function. Let

$x_1, ..., x_n$ be n measurements of that distribution. With a large n, we can give an accurate estimate of the mean value μ by computing the mean of the measurements $\frac{x_1+x_2+...+x_n}{n}$ regardless of the covariance σ.

Now let us assume we are instead provided with the values $x_2 - x_1, x_3 - x_1, ..., x_n - x_1$. The mean value of these measurements converges to the error of x_1.

Indeed: $\frac{(x_2-x_1)+(x_3-x_1)+...+(x_n-x_1)}{n} = \frac{x_2+...+x_n}{n} - x_1 \simeq \mu - x_1$. In other words, when corrupting all measurements with the first measurement, we worsened our estimation of the actual mean of the distribution function.

From now on, we will only consider this reformulated closed-form solution. We refer to it as standard closed-form solution.

4.3 Impact of bias

The term bias means a **systematic** error in the measurements. Let us assume we have a sensor measuring a physical quantity μ. In the case of an ideal sensor, the measurements provided by the sensor are equal to μ.

However, in real applications, sensors are not ideal. In robotics we often assume that a sensor provides a measurements affected by a Gaussian noise. In particular, each measurement is distributed according to a normal distribution function $N(\mu - B, \sigma)$. The covariance σ determines the non-systematic error, it is inversely proportional to the precision of the sensor. The term B refers to the bias. Since this is a systematic error, if we were to take the mean of n measurements as we did in Section 4.2 we would end up estimating the quantity $\mu - B$ instead of μ.

To make computations easier, the bias is often considered constant with respect to time. In general this is not the case. For instance, an IMU bias will vary over time as the IMU overheats. However, it is a reasonable assumption to consider it constant over a time interval shorter than a minute even for a low cost IMU.

Since the closed-form solution runs over a significantly shorter time interval of around 3 seconds, it is reasonable to assume the gyroscope bias and the accelerometer bias constant.

It is reasonable to account for the bias without accounting for covariance σ. By definition, if we can subtract the bias from our measurements, then the remaining errors have a zero-mean. Therefore, integrating these measurements over a time interval yields a statistically negligible error if provided with enough measurements. This assumption is reversed in the presence of bias. Since it is integrated, even the smallest constant bias can make the error grows to infinity. This phenomenon is known as drift.

4.3.1 Accelerometer bias

In [18], the author provides an alternative formulation of the closed-form solution including the accelerometer bias as an observable unknown of the system.

In other words, the definition of the accelerometer measurements given in Equation 3.2 becomes:

$$A_\tau(\tau) = A_\tau^{inertial}(\tau) - G_\tau + B$$

With B the accelerometer bias. We therefore have an additional term in Equation 3.4:

$$r(t) = r(T_{in}) \; + \; v(T_{in})\Delta_t \; + \; g\frac{\Delta_t^2}{2} \; + \; C^{T_{in}} S_{T_{in}}(t) + C^{T_{in}} \Gamma(t) B$$

Where:

$$\Gamma(t) = \int_{T_{in}}^{t} (t-\tau) C_{T_{in}}^{\tau} \mathrm{d}\tau$$

The equation yielding the linear system is therefore identical to the previous one in 3.9 with the new term $\Gamma_j = \Gamma(t_j)$ that multiplies the accelerometer bias:

$$S_j = \lambda_1^i \mu_1^i - V t_j - G\frac{t_j^2}{2} - \lambda_j^i \mu_j^i + \Gamma_j B$$

To take this new term into account, we have to tweak our linear system. Since the accelerometer bias B is an unknown, we insert it into the vector X. Moreover, we insert at the same indexes three additional columns in Ξ (recall that Γ_j is a 3D vector): $[\Gamma_2, ..., \Gamma_2, \Gamma_3, ...\Gamma_3, ..., \Gamma_{n_i}]^T$ The vector S remains unchanged.

Figure 4.3 represents the estimation of the accelerometer bias with respect to the duration of integration.

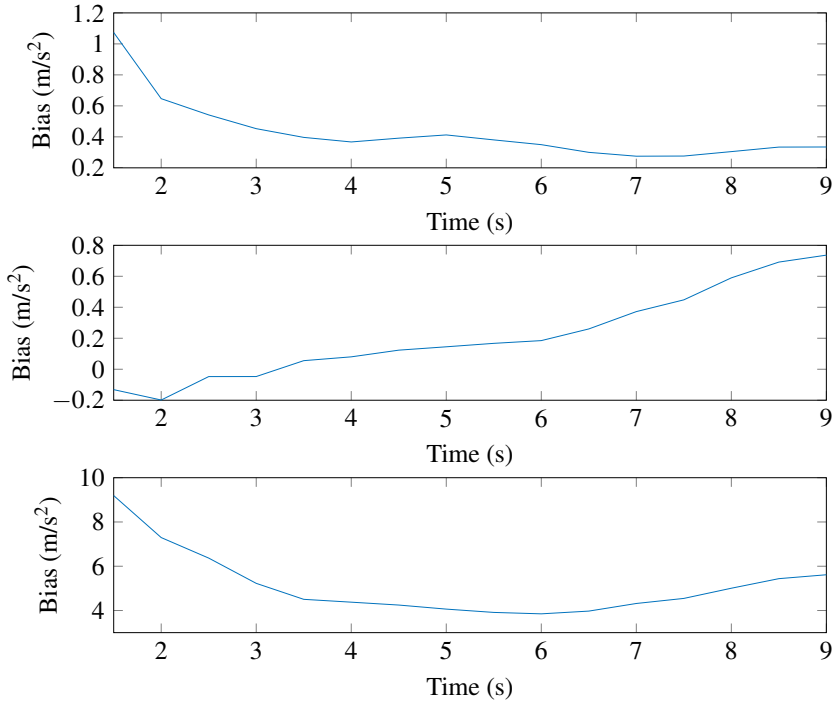
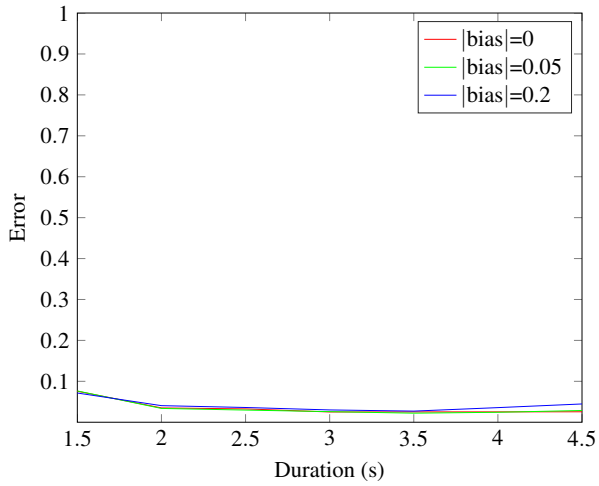


Figure 4.3 – Accelerometer bias estimation from closed-form solution. We are observing 7 features over a variable duration of integration.

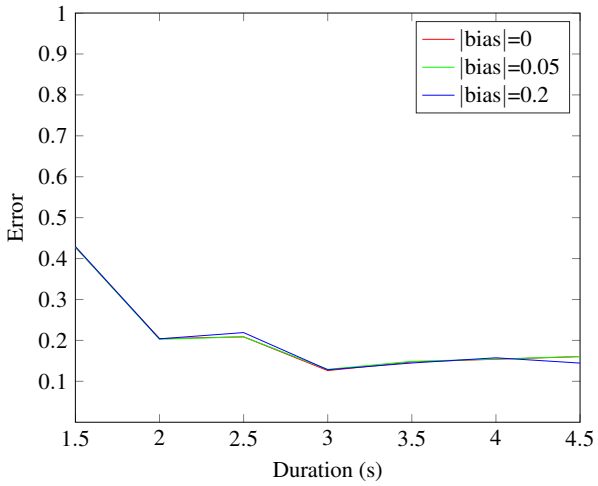
As seen in Figure 4.3, the estimation of the accelerometer bias with this method is not robust. Indeed, we noticed that the accelerometer bias estimation does not converge for large duration of integration.

The reason why the accelerometer bias estimation is not robust is because our system is only slightly affected by it. In other words, despite a high accelerometer bias the closed-form solution still provides robust results.

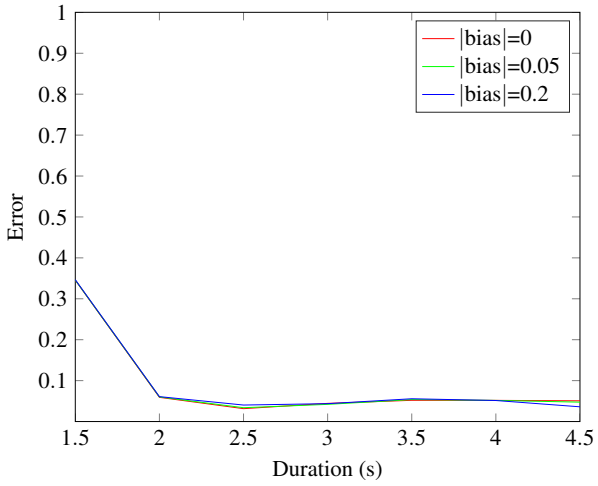
In order to visualize this, we simulated high accelerometer bias by artificially increasing it from the data provided by our terrain IMU (Figure 4.4).



(a) Gravity estimation error



(b) Velocity estimation error



(c) Lambda estimation error

Figure 4.4 – Impact of the accelerometer bias on the performance of the closed-form solution. We are observing 7 features over a variable duration of integration.

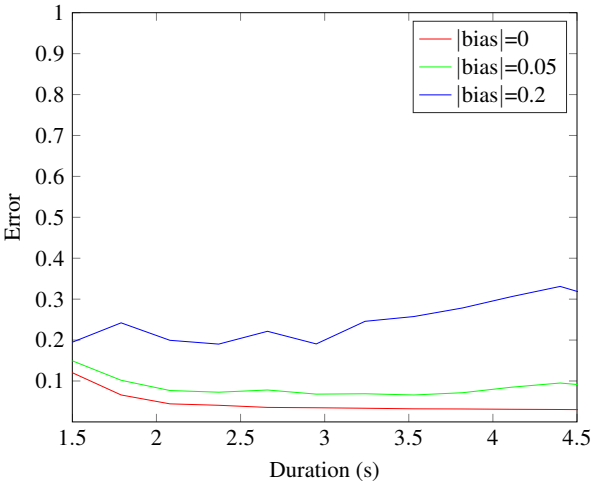
As seen on the Figure 4.4, neither the gravity, the velocity or the lambdas are impacted by the accelerometer bias.

This is a counterintuitive results. Since our equations contain an integration of the acceleration, we also perform an integration of the accelerometer bias (see Equation 3.5). We would have expected the accelerometer bias to have a greater impact on the solutions yielded by the system.

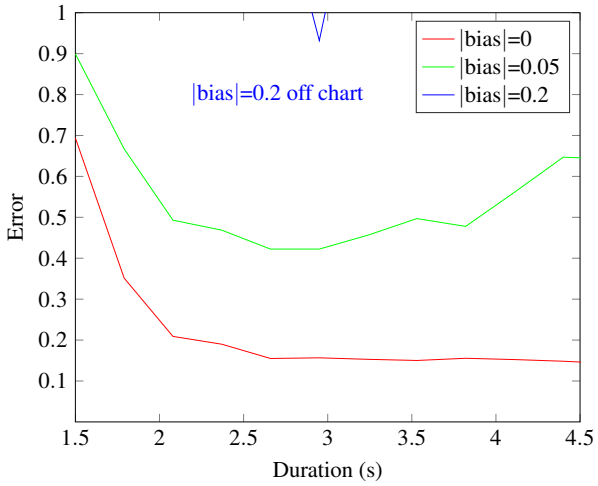
4.3.2 Gyroscope bias

Our experiments have shown that the presence of gyroscope bias significantly damages the results of the closed-form solution.

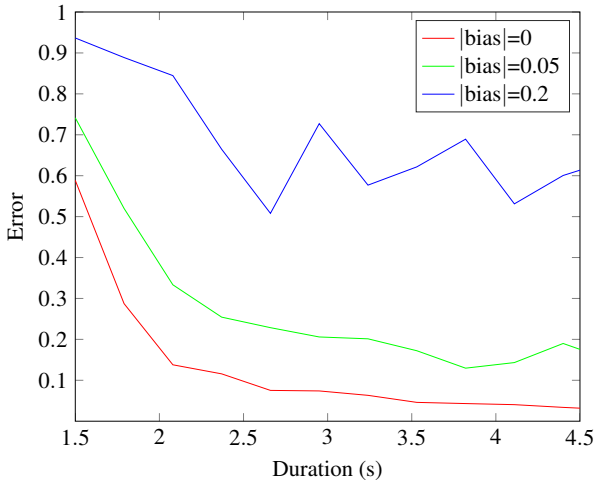
Again, in order to visualize this, we artificially corrupt the true gyroscope measurements by adding a bias (Figure 4.5).



(a) Gravity estimation error



(b) Velocity estimation error



(c) Lambda estimation error

Figure 4.5 – Impact of the gyroscope bias on the performance of the closed-form solution. We are observing 7 features over a variable duration of integration.

As seen in Figure 4.5, the performance becomes very poor in presence of a bias on the gyroscope and, in practice, the overall method could only be successfully used with a very precise—and expensive—gyroscope.

Previous work has shown that the gyroscope bias is an observable mode when using an IMU and a camera, which means that it can be estimated [19].

Previously, we said that the rotation matrix $C_{T_{in}}^{t_j}$ was recovered from the gyroscope measurements but we did not explicitly give its expression. We can compute it by using the following formulas:

$$\begin{cases} C_{T_{in}}^{T_{in}} &= I_3 \\ C_{T_{in}}^{t_i} &= C_{T_{in}}^{t_{i-1}}(I_3 + skew(\Omega_{i-1})(t_i - t_{i-1})) \end{cases}$$

Where:

- t_i is the time where the i$^{\text{th}}$ gyroscope measurements occurred, starting at $t_0 = T_{in}$;
- Ω_i is the true angular velocity;
- $skew(v)$ is the function that returns the skew symmetric matrix corresponding to the vector v

The gyroscope provides the angular velocity with a bias:

$$\Omega_i^{gyro} = \Omega_i + B$$

Where B is the gyroscope bias and Ω_i^{gyro} is the angular velocity measured by the gyroscope. Note that we can consider the bias constant over time since our integration time is short. Let us factor out the gyroscope bias in the computation of the rotation matrix $C_{T_{in}}^{t_i}$:

$$\begin{aligned} C_{T_{in}}^{t_i} &= C_{T_{in}}^{t_{i-1}}(I_3 + skew(\Omega_{i-1}^{gyro} - B)(t_i - t_{i-1})) \\ &= C_{T_{in}}^{t_{i-1}}(I_3 + skew(\Omega_{i-1}^{gyro})(t_i - t_{i-1}) - skew(B)(t_i - t_{i-1})) \\ &= C_{T_{in}}^{t_{i-1}}(I_3 + skew(\Omega_{i-1}^{gyro})(t_i - t_{i-1})) - C_{T_{in}}^{t_{i-1}} skew(B)(t_i - t_{i-1}) \end{aligned}$$

The rotation matrix $C_{T_{in}}^{t_i}$ is computed from multiplications of previously computed rotation matrix $C_{T_{in}}^{t_{i-1}}$. This process is called an exponentiation, and unfortunately we cannot express the gyroscope bias linearly from this method. One could use a Taylor expansion to have a linear approximation of the exponentiated gyroscope bias. However, since this rotation matrix multiplies other unknowns in Equation 3.9, the system would still not be linear. It is therefore not possible to add the gyroscope bias in our unknown vector X and determine X by simply inverting a matrix as in the standard closed-form solution .

In the next chapter, we propose a different approach to estimate the gyroscope bias using the closed-form solution.

— 5 —
Estimating the gyroscope bias

5.1 Previous methods

Estimating the gyroscope bias offline is trivial. Indeed, when the gyroscope is at rest, the angular velocity is null. Therefore, a gyroscope at rest directly provides its bias. Some applications use this technique and stop all actuators of the robots periodically to recalibrate the gyroscope. However, a MAV has to maintain itself in the air and does not have such zero-position.

Constructors sometimes provide a table of the gyroscope bias with respect to its temperature. Most often this pre-calibrated bias is not accurate, since the bias value does not only depend on the temperature. A robust estimation of the gyroscope bias should be done online, when the platform is in motion.

There are several methods to estimate the gyroscope bias online. A nonlinear filter that estimates the gyroscope bias using just an IMU is proposed in [20], but its convergence requires the MAV to have a specific motion. They can get rid of this constraint by adding a magnetometer on the platform.

A standard method when performing a Kalman Filter is to add the bias to the filter state [27] [24]. This method seems to converge to accurate values of the bias even for arbitrary initialization to 0. However, increasing the size of the estimated state can make the estimation process less accurate, especially when the filter is provided with erroneous initial state. A different approach is proposed in [25] where the bias is estimated separately in the first place. This technique only relies on inertial measurements. Therefore, it requires a long time for its convergence, more than 100 seconds for a robust gyroscope bias estimation.

In the following section, we will present an innovative way to estimate the gyroscope bias with the closed-form solution. It can robustly estimate high gyroscope bias and requires short time of integration of the order of 2 seconds.

5.2 Estimating the gyroscope bias with the closed-form solution

We stated earlier that the gyroscope bias was the bottleneck of the closed-form solution. In other words, if we find the correct gyroscope bias, then the linear system $\Xi X = S$ admits a proper solution. What we call proper solution here, is a solution that satisfies all the equations with a low residual. Indeed, since our system is overconstrained (there are more equations than

unknowns), it is unlikely to find a solution satisfying all the equations exactly. We instead find the solution X that minimizes the residual $|\Xi X - S|^2$.

Since our system is linear, the solution X minimizing the residual can be expressed in closed-form using the pseudo-inverse. However, as seen in Section 4.3.2, the gyroscope bias can not be expressed linearly. Therefore, we use an alternative optimization method to minimize the residual $|\Xi X - S|^2$ with respect to the gyroscope bias. Specifically, this is a non-linear least square minimization problem, which can be solved by iterative gradient descent.

In other words, we minimize the cost function $c(B) = |\Xi X - S|^2$, with Ξ and S computed according to the bias on the gyroscope B. We can initialize this optimization with $B = 0_3$ since the bias is usually a rather small quantity. This technique is similar to the one used in [1], where the author minimizes the residual of a Kalman filter in order to recover the noise matrix for the sensors.

Unlike the estimation of the accelerometer bias, we found that this method converges with respect to the duration of the integration. Moreover, the estimation of the gyroscope bias is accurate. In Figures 5.1 and 5.2 we plot the quality of our evaluations with respect to the duration of the integration. We observe 30 point features and the gyroscope bias is artificially set to $[0.0276, -0.0024, 0.0417]$ rad/s such that its norm is 0.05, which is a viable bias value for a cheap gyroscope in operation.

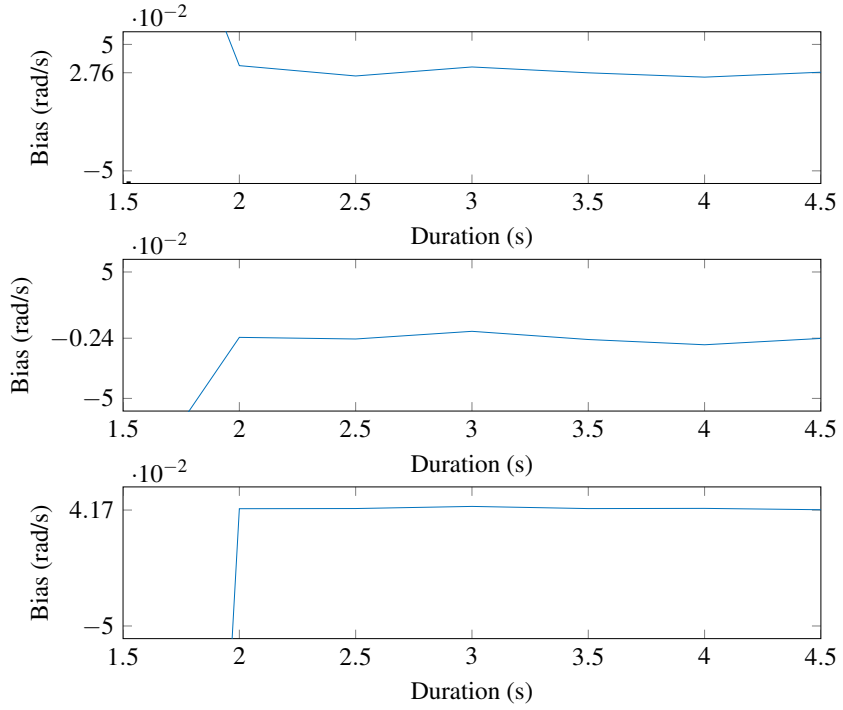


Figure 5.1 – Gyroscope bias estimation from non-linear minimization of the residual. We are observing 30 features over a variable duration of integration. The true gyroscope bias is $[0.0276, -0.0024, 0.0417]$.

The optimized closed-form solution provides better results than the standard closed-form solution. Figure 5.2 depicts an improvement in precision of around 5% for the distance to the features, and around 13% for the speed after 4 seconds of integration.

Moreover, the optimized closed-form solution requires a shorter integration duration to provide good quality results. Specifically, after 2 seconds of integration (around 7 camera

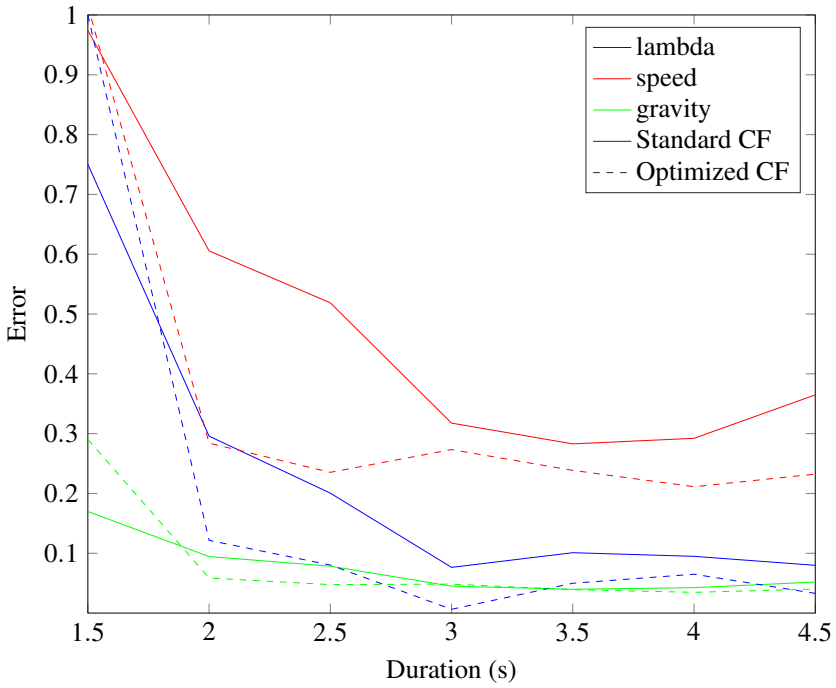


Figure 5.2 – Estimation error of the closed-form solution against the optimized closed-form solution. We are observing 30 features over a variable duration of integration. The true gyroscope bias is $[0.0276, -0.0024, 0.0417]$.

observations) the provided estimations are already robust. The non-optimized closed-form solution requires 3 seconds of integration before converging to acceptable estimations.

Lastly, this method is robust even for high values of the gyroscope bias. Figure 5.3 represents the quality of the estimations with the same artificial gyroscope bias from Figure 4.5.

As seen in Figure 5.3, after a certain integration duration, the estimations agree no matter how high the bias is. Moreover, the values on which these evaluations agree are equals to the ones in Figure 4.5 for the curve that has a gyroscope bias set to 0.

By adding this new method for the bias estimation to the original method we obtain results which are equivalent to the ones in absence of bias. Compared to the original method, the new method is now robust to the gyroscope bias, and also provides the gyroscope bias.

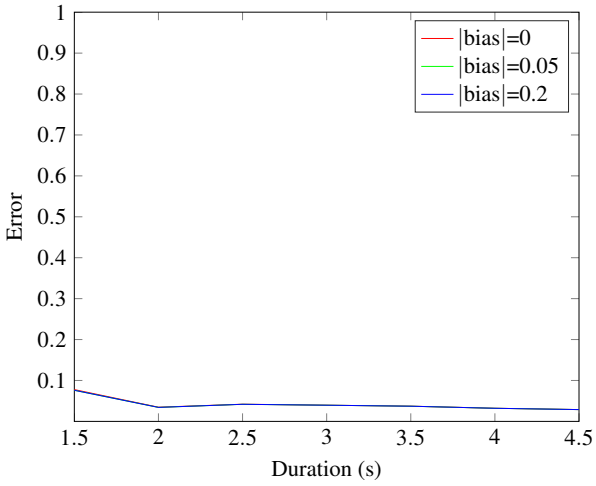
In the following sections we will discuss the limitations of this method.

5.3 Optimization with low amount of measurements

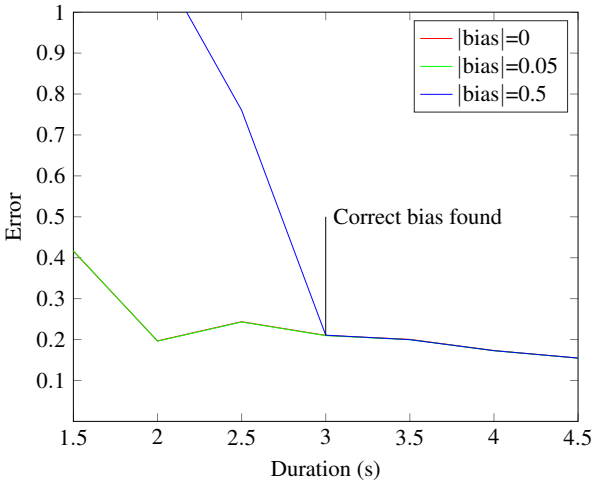
For the previous experiment, the number of observed features was high. However, for a lower amount of observed features, the optimization provide poor estimations for short integration duration.

In Figures 5.4 and 5.5 we plot the quality of our evaluations with respect to the duration of the integration. We observe 7 features and the gyroscope bias is still artificially set to $[0.0276, -0.0024, 0.0417]$.

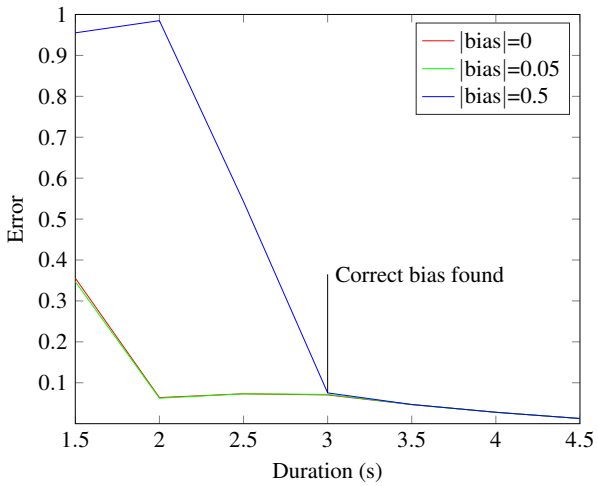
As seen in Figure 5.5, for long integration duration (> 2.5 seconds), we obtain similar results with the optimization regardless of the number of features. Indeed, in this case, the gyroscope bias is still robustly estimated as seen in Figure 5.4.



(a) Gravity estimation relative error



(b) Velocity estimation relative error



(c) Lambda estimation relative error

Figure 5.3 – Impact of the gyroscope bias on the performance of the optimized closed-form solution. We are observing 7 features over a variable duration of integration.

However, for short integration duration (< 2.5 seconds) the optimization incorrectly estimates gyroscope bias to large values. This misestimation leads the optimized closed-form solution to provide worse results than the standard closed-form solution for short integration duration when observing a small amount of features.

To understand this misestimation, in Figure 5.6 we plot the residual with respect to the bias, which is the cost function we are minimizing. We highlight a misestimation of the gyroscope bias by setting the duration of integration to 2 seconds while observing 7 features. We refer to the components of the gyroscope bias by $B = [B_x, B_y, B_z]$.

As we can see in Figure 5.6, the cost function strongly depends on the two components B_x and B_y. Indeed, for values of B_x and B_y far from the actual bias (0.0276 and -0.0024 respectively), the residual is high. In other words, a gradient descent will correctly find the actual bias components B_x and B_y.

On the other hand, B_z does not affect the residual much. Indeed, for values B_z far from the actual bias (0.0417), the residual is only slightly affected. Moreover, the minimum of the cost

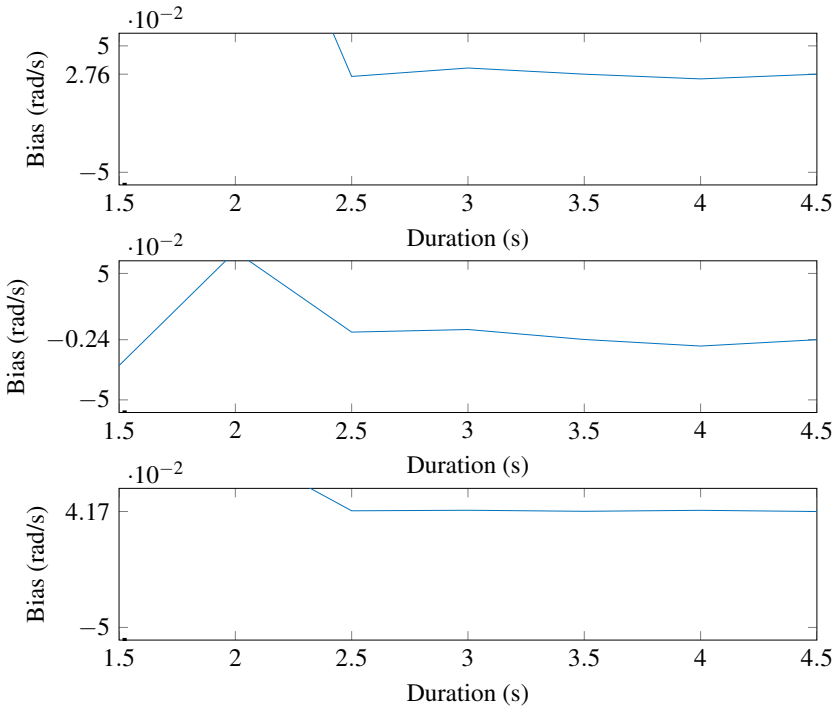


Figure 5.4 – Gyroscope bias estimation from non-linear minimization of the residual. We are observing 7 features over a variable duration of integration. The true gyroscope bias is $[0.0276, -0.0024, 0.0417]$.

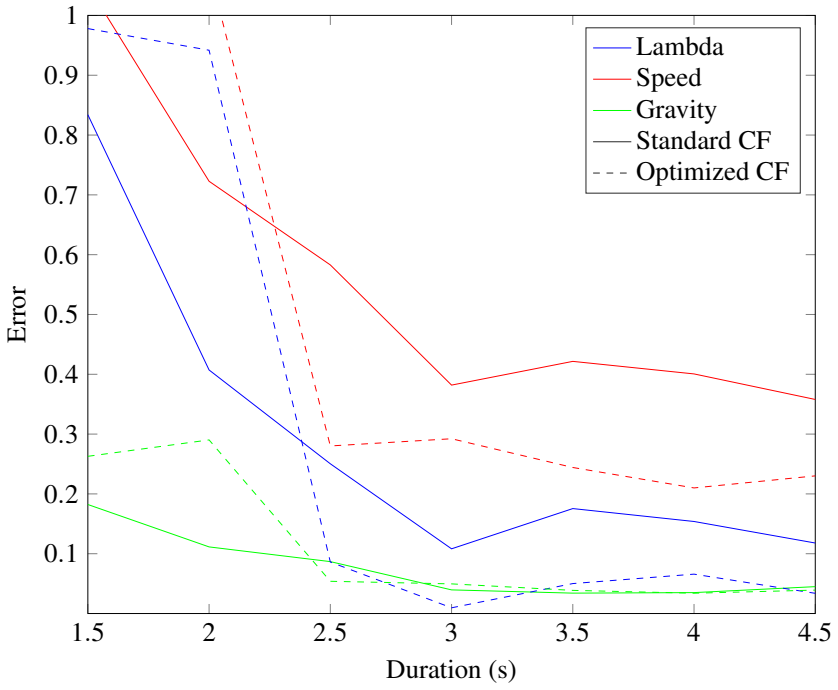
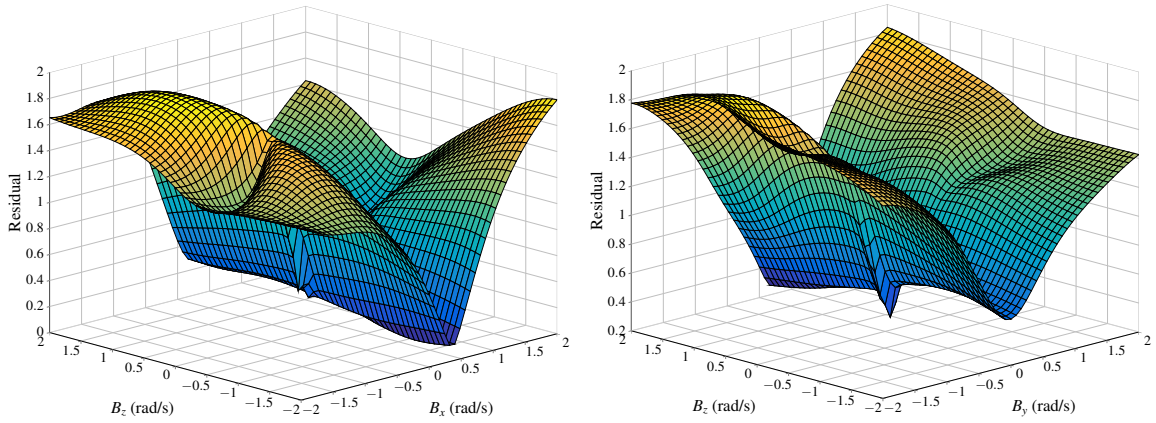


Figure 5.5 – Estimation error of the closed-form solution against the optimized closed-form solution. We are observing 7 features over a variable duration of integration. The true gyroscope bias is $[0.0276, -0.0024, 0.0417]$.



(a) Residual with respect to B_x and B_z

(b) Residual with respect to B_y and B_z

Figure 5.6 – Cost function (residual) with respect to the gyroscope bias for a small amount of available measurements (integration of 2 seconds while observing 7 features)

function is reached at $|B_z| > 1$ rad/s which is a very high and unlikely value for a gyroscope bias.

However, we also notice in Figure 5.6 a local minimum at the correct value of B_z. The reason why this minimum is not reached is because there are other values of B_z that further minimizes the cost function.

These plots raise two issues we will address in the following sections:

- Why is the cost affected differently by the components of the gyroscope bias;
- How can we force our method to correctly estimate the bias when we only have a low amount of measurements (short integration duration and small number of features).

5.3.1 Residual with respect to the 3 components of B

In Figure 5.6 we realized that our cost function was affected differently by B_x, B_y and B_z. Specifically, the residual was almost constant with respect to B_z. In this case, the cost function admits a symmetry with respect to B_z.

We were able to replicate this situation with simulated motions. We found that the residual was almost constant with respect to the component of the gyroscope bias along the direction $\vec{u}$ when this direction $\vec{u}$ is collinear with the gravity throughout the motion.

We represent this case in Figure 5.7. In the terrain data we had, the motion satisfies this constraint. Specifically, the gyroscope was strapped on the MAV such that the vector $[0,0,1]$ in the gyroscope frame was pointing upwards when the MAV was hovering. That is why the residual varies only slightly for certain time sequences with respect to this vector. Indeed, in normal operations, a MAV will often have a pose close to its hovering stance in order to stay stable.

If the MAV rotates such that the vector $\vec{u}$ becomes non collinear with the gravity, the cost function does not exhibit this symmetry anymore. Hence the gradient descent can work correctly.

A simple solution to avoid having that symmetry in our system would be to constrain the motion of our MAV while it is operating. In the following section, we introduce a modification

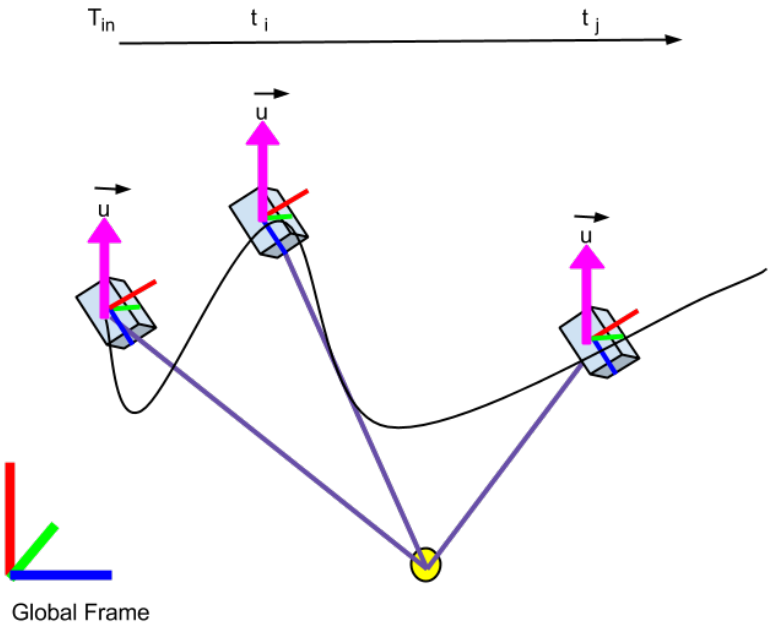


Figure 5.7 – Case when the cost function is constant with respect to the component of the gyroscope bias along the direction $\vec{u}$.

to the cost function in order to artificially get rid of this symmetry, preventing the estimation of the bias to reach highly unlikely values.

5.3.2 Regularization term

We introduce a simple modification to the cost function to improve the estimations provided by the optimized closed-form solution when the amount of available measurements is low.

The modification relies on the assumption that the gyroscope bias is close to 0, and the fact that our cost function admits a local minima close to the actual gyroscope bias.

We add to our cost function a term that penalizes high gyroscope bias. In statistics, this is often called a regularization term. Specifically, the cost function becomes:

$$c(B) = |\Xi X - S|^2 + \lambda |B|$$

The coefficient λ is the weight given to how much we want the bias to be small. For small values of λ, our cost function is similar to the previous one and the bias can grow arbitrarily high. For high values of λ, the estimations provided by the optimized closed-form solution are similar to the ones provided by the standard closed-form solution. Indeed, high values of λ force the estimation of the bias to 0. Therefore, it is not compensated when computing the rotation matrices $C_{T_{in}}^{t_i}$. In other words, the computed matrices Ξ and S will be equal to the the ones in the standard closed-form solution.

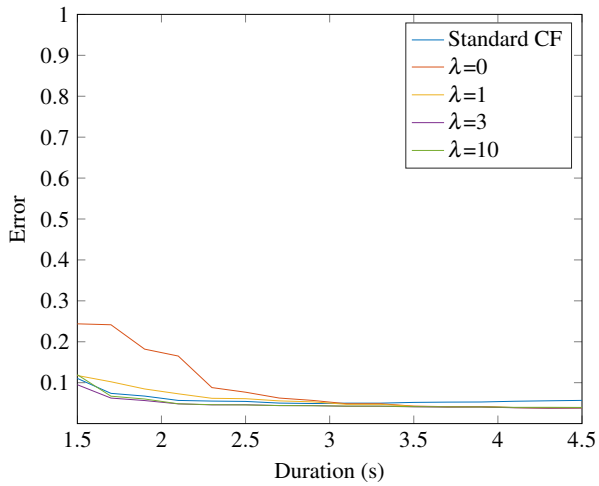
Note that, instead of forcing the gyroscope bias to be close to 0, we can easily force it to be close to any value. Therefore, if we have the knowledge of an approximately known gyroscope bias, we can use it to provide a better estimation of the gyroscope bias.

$$c(B) = |\Xi X - S|^2 + \lambda |B - B^{approx}|$$

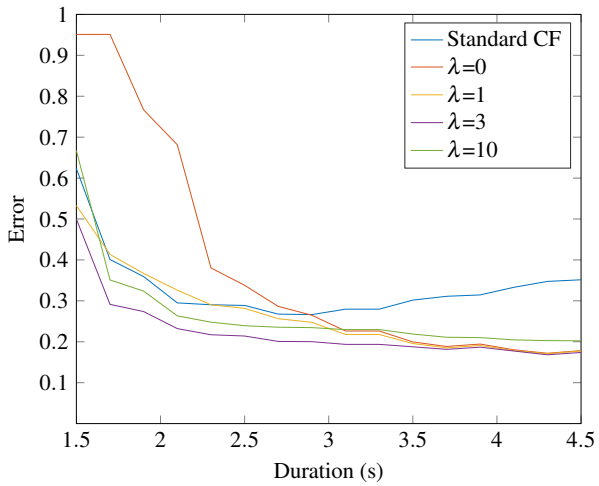
With B^{approx} the known approximate gyroscope bias. This methods allow us to reuse previously computed gyroscope bias since it is known to slowly vary over time.

Selecting a reliable and safe value for the regularization parameter λ is complicated. In this report, we will pick a value of λ by experimentation.

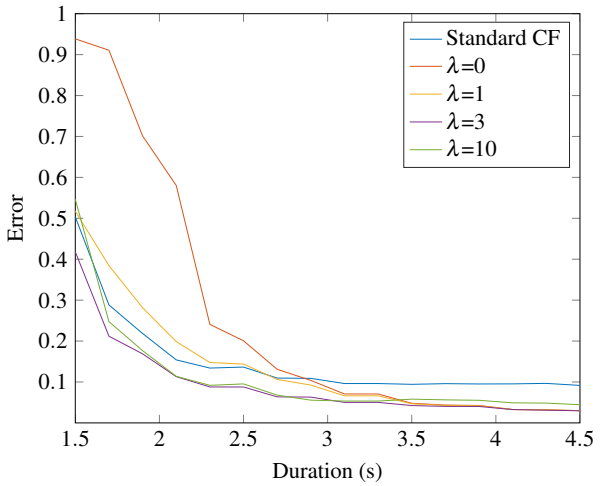
Figure 5.8 represents the error of our estimations with respect to the integration duration, with different regularization parameters. In order to reduce overfitting our choice of λ to a specific sample, we considered a set of motions with randomized realistic gyroscope bias and took the mean error.



(a) Gravity estimation relative error



(b) Velocity estimation relative error



(c) Lambda estimation relative error

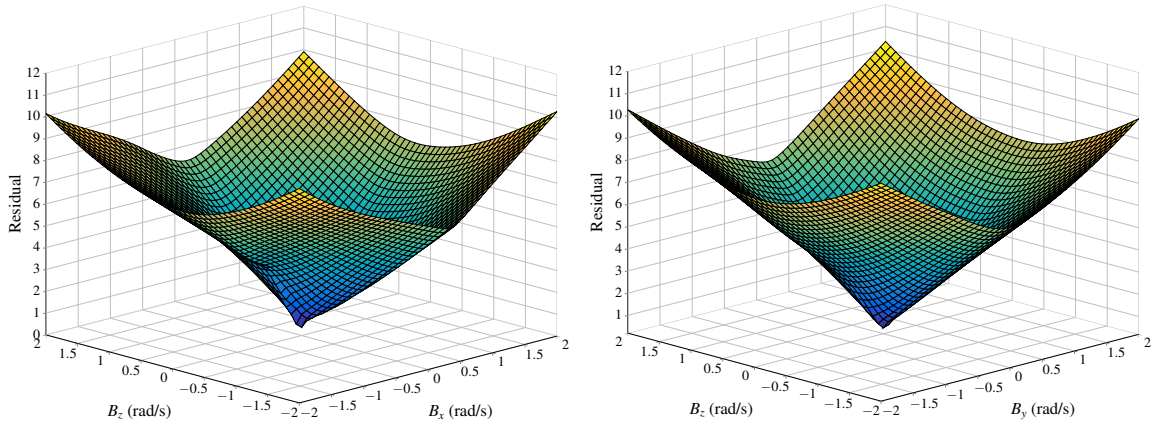
Figure 5.8 – Estimation error of the optimized closed-form solution with different regularization parameters. We are observing 7 features over a variable duration of integration.

Note that $\lambda = 0$ is equivalent to having no regularization term. As expected, the curves in Figure 5.8 with small values of λ require a longer duration of integration to robustly estimate the state. The curves for larger values of λ provide equivalent estimations than the standard

closed-form solution without the optimization process. Indeed, these curves estimate the gyroscope bias to small values thus almost not compensating for it.

Note that, as the integration duration increases, the estimations agree regardless of the regularization parameter. Indeed, adding more observations grows the size of the residual $|\Xi X - S|$ thus implicitly decreases the weight of the regularization parameter in the cost function. This effect is much wanted since the purpose of the regularization parameter is to regulate the estimation of the bias when the amount of available measurements is low.

In Figure 5.9 we plot our tweaked cost function for an integration duration of 2 seconds. This highlights how the regularization parameter fixes the misestimation of the gyroscope bias for low amount of measurements.



(a) Residual with respect to B_x and B_z

(b) Residual with respect to B_y and B_z

Figure 5.9 – Regularized cost function with respect to the gyroscope bias for a small amount of available measurements (integration of 2 seconds while observing 7 features). Regularization parameter is $\lambda = 3$.

As expected, the actual gyroscope bias is now the global minima since we penalized unlikely high bias values. Since the optimization starts at 0 we are confident that this method will converge correctly. Figures 5.10 and 5.11 represents the quality of our estimations with low amount of measurements using a regularization parameter $\lambda = 3$.

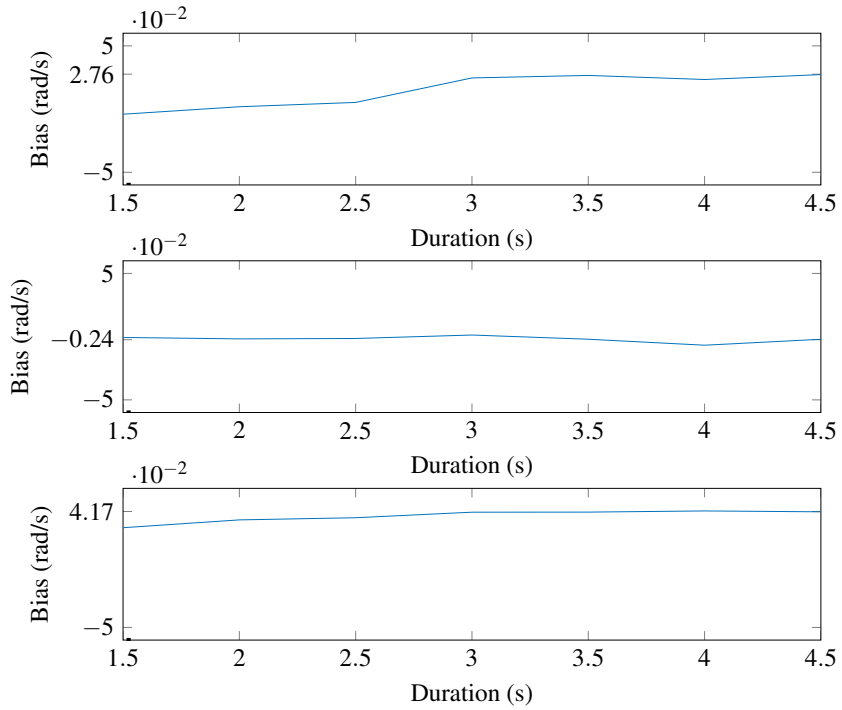


Figure 5.10 – Gyroscope bias estimation from non-linear minimization of the residual with a regularization term $\lambda = 3$. We are observing 7 features over a variable duration of integration. The true gyroscope bias is $[0.0276, -0.0024, 0.0417]$.

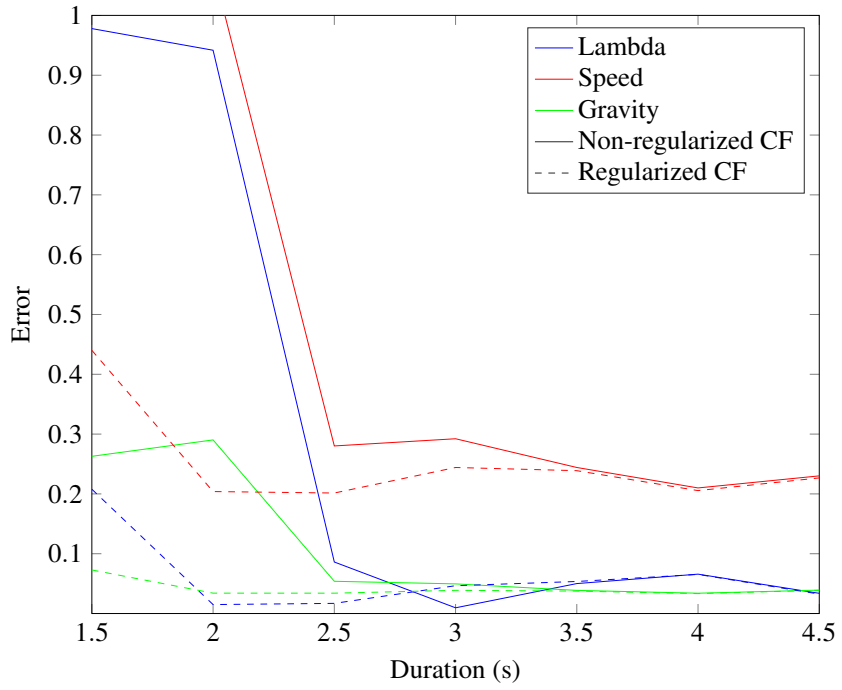


Figure 5.11 – Estimation error of the regularized ($\lambda = 3$) closed-form solution against optimized **non-regularized** closed-form solution. We are observing 7 features over a variable duration of integration. The true gyroscope bias is $[0.0276, -0.0024, 0.0417]$.

— 6 —
Validation

In this chapter we compare the performance on the estimations of the gravity, the initial velocity and the absolute scale obtained with three different methods:

- Our modified closed-form solution;
- The original closed-form solution [18];
- The Kalman filter described in [26].

The reason we included a Kalman filter in the validation is for having a reference to state of the art fusion method. This filter has been used as the sole sensor fusion algorithm on a MAV for autonomous navigation in unknown environments. It fuses visual and inertial measurements in order to recover the speed and the gravity in the local frame (equivalent to absolute roll and pitch angles). The visual information is processed separately as a pose-estimator up to scale that is then fed to the sensor fusion process. Specifically, this filter uses the parallel localization and mapping algorithm (PTAM) provided with a good initial guess of the absolute scale [12].

We set the integration duration for the closed-form solution to 3 seconds. The camera provides 30fps, but we discard most of the observations and consider only one observation every 0.3 seconds. Indeed, adding observations that are too close to each other does not add significant information to our system.

Since the gyroscope bias varies slowly over time, we can reuse its knowledge. In other words, we can use the optimized closed-form solution to compute the bias at the beginning of the sequence. Thereafter, we can quickly provide reliable state estimations with the standard closed-form solution by compensating with the computed bias.

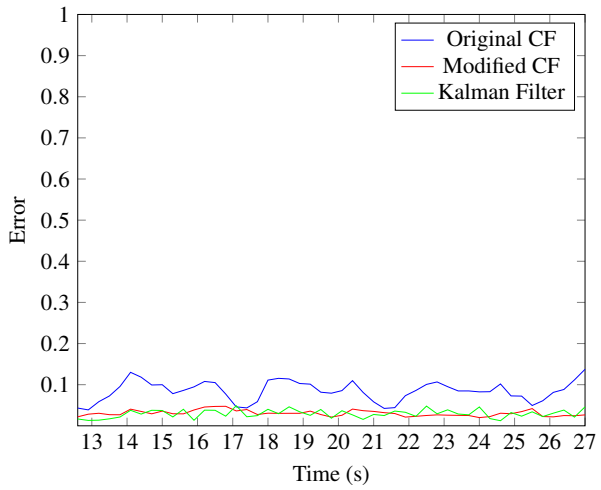
For long term navigation (longer than a minute), it would be advised to recompute the gyroscope bias every so often. This re-computation can use the knowledge of previously computed bias as stated in Section 5.3.2.

The reason why we do not run the optimization for every state estimation is to speed up the computation. Despite the optimization is relatively fast since our cost function is defined only with respect to the gyroscope bias, it still has to run the closed-form solution several times for its gradient descent to converge. We found by experimentation that it usually requires around 6 iterations to converge.

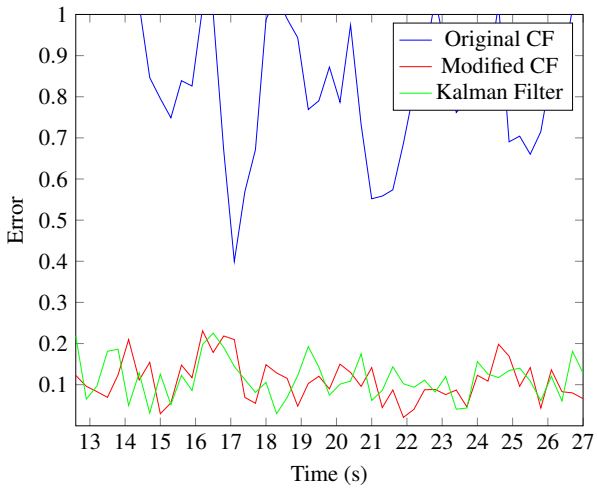
Optimally, we would have validated our method against a different dataset than the one we used in the previous sections to draw our conclusions. Indeed, using the same dataset does not prove that our conclusions generalize well to other motions and measurements (overfitting). However, such datasets are rare due to the expensive equipment required to acquire the ground

truth (optical Vicon tracking system). Unfortunately, we have not been able to obtain a second dataset in time for a proper validation. Therefore, the dataset used for validation is the same as the one we used in the previous sections. Specifically, it contains IMU measurements and ground truth, but the camera observations are simulated.

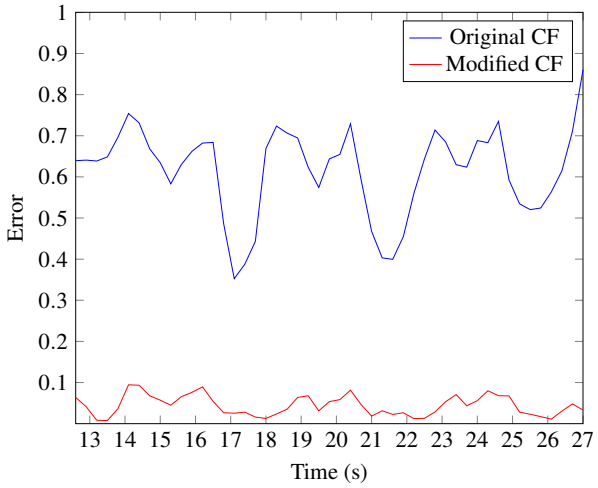
The operation starts at time $T_{in} = 12.6$ seconds and ends at time $t = 30$ seconds.



(a) Gravity estimation relative error



(b) Velocity estimation relative error



(c) Lambda estimation relative error

Figure 6.1 – Estimation error of the regularized optimized closed-form solution (with regularization paramater $\lambda = 3$) against the original closed-form solution [18] and Weiss' Kalman filter [26]. The duration of integration is set to 3 seconds, and 16 point features are observed throughout the whole operation. The gyroscope bias is estimated by the modified closed-form solution to $[0.0029, 0.0046, -0.0196]$.

As seen in Figure 6.1, the modified closed-form solution performs much better than the original closed-form solution. What is more, the modified closed-form solution has equivalent performance than Weiss' Kalman filter. We remind the reader that unlike Weiss' Kalman filter, the modified closed-form solution does not require an initial state to be provided. The gyroscope bias has been estimated by the modified closed-form solution to $[0.0029, 0.0046, -0.0196]$

the first time we ran the estimation process at $T_{in} = 12.6$ seconds. We reused this estimated bias for the subsequent estimations, allowing us to skip the optimization process.

— 7 —
Conclusion

For a MAV, limiting the number of on-board sensor is important to save power consumption and processing power. A popular choice of sensors is a camera coupled with an IMU for their complementary. However, the methods for fusing visual and inertial measurements so far introduced are filter based, hence require an initialization. Providing a reliable state initialization is critical for these algorithms to work correctly.

In this report, we have studied the recent closed-form solution proposed by [18] that performs visual-inertial sensor fusion without requiring an initialization. We implemented this method in order to test it with real terrain data. This allowed us to identify its performance bottlenecks and bring modifications to overcome them.

Specifically, we started by reformulating the equations for greater numerical stability. This lead to a major leap in the quality of the estimations.

We then investigated the impact of biased inertial measurements. Despite the case of biased accelerometer was originally studied in [18] we show that its low impact on the system makes it hard to estimate.

One major performance bottleneck of this method was the impact of biased gyroscope measurements. In other words, the performance becomes very poor in presence of a bias on the gyroscope and, in practice, the overall method could only be successfully used with a very precise - and expensive - gyroscope. We then introduced a simple method that automatically estimates this bias.

However, this method requires a significant amount of data to provide correct estimates of the gyroscope bias (either many features or long integration duration). We were able to get rid of this limitation with a simple statistical trick by adding a regularization term to our cost function.

By adding this new method for the bias estimation to the original method we obtain results which are equivalent to the ones in absence of bias. Compared to the original method, the new method is now robust to the gyroscope bias, and also provides the gyroscope bias.

We validated this method by comparing its performance against the original method and the Extended Kalman Filter described in [26] which is the state of the art approach for visual-inertial sensor fusion on MAV.

For future work, we see this optimized closed-form solution being implemented on a MAV to provide accurate state initialization. This would allow aggressive take-off manoeuvres, such as hand throwing the MAV in the air or parachuting it from a plane. Indeed, these manoeuvres are not currently performed because of the lack of a proper initialization method.

IMU Camera transformation

Thus far, we assumed that the camera and the IMU were a single particle in space. We made this assumption to ease the derivation of the equations.

However, we have to slightly modify our equations to take into account the IMU camera transformation. Indeed, in a real application, camera and IMU are different sensors and are therefore strapped at different position on the drone. It is assumed that we know the calibration (translation/rotation) between the IMU frame and the camera frame.

We have the choice between:

1. Setting the local frame to the IMU frame;

2. Setting the local frame to the camera frame.

In robotics, the former is usually preferred. Indeed, note that the later requires more computations. By definition, the angular velocity is identical on all the point of a rigid body, so it is easy to use gyroscope measurements in the camera frame. However, it is not the case for linear acceleration. Specifically, we have to use the angular velocity to transform the acceleration from a point to another on the rigid body, thus introducing more errors in the system.

We will therefore express the camera measurements with respect to the IMU frame. The previous equations we derived up to Section 3.4 are still valid when considering only the IMU frame (there is no more coincided frame). In other words, $r(t)$ now refers to the position of the IMU.

Let us re-express the bearing Equation 3.6 with respect to the IMU frame.

$$p^i = r(t) + C^{T_{in}} C^{IMU}_{Cam} C^t_{T_{in}} P^i_t(t) - C^{T_{in}} C^{IMU}_{Cam} C^t_{T_{in}} T^{IMU}_{Cam} \tag{7.1}$$

Where T^{IMU}_{Cam} is the translation from the camera frame to the IMU frame expressed in the camera frame, and C^{IMU}_{Cam} is the rotation from the camera frame to the IMU frame. These two terms are given from the IMU camera calibration.

Moreover, for the first observation at time $t = T_{in}$ we now have the following equation instead of 3.7:

$$p^i - r(T_{in}) = C^{T_{in}} C^{IMU}_{Cam} P^i_{T_{in}}(T_{in}) - C^{T_{in}} C^{IMU}_{Cam} T^{IMU}_{Cam} \tag{7.2}$$

We can inject the equation of the position of the platform 3.1 into our new equation of a bearing and simplify with the equation of the initial bearing:

$$
\begin{aligned}
& C^{T_{in}} C^{IMU}_{Cam} P^i_{T_{in}}(T_{in}) - C^{T_{in}} C^{IMU}_{Cam} T^{IMU}_{Cam} = \\
& v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + C^{T_{in}} S_{T_{in}}(t) + C^{T_{in}} C^{IMU}_{Cam} C^t_{T_{in}} P^i_t(t) - C^{T_{in}} C^{IMU}_{Cam} C^t_{T_{in}} T^{IMU}_{Cam} \\
\Leftrightarrow \quad & C^{T_{in}} C^{IMU}_{Cam} P^i_{T_{in}}(T_{in}) = \\
& v(T_{in})\Delta_t + g\frac{\Delta_t^2}{2} + C^{T_{in}} S_{T_{in}}(t) + C^{T_{in}} C^{IMU}_{Cam} C^t_{T_{in}} P^i_t(t) - C^{T_{in}} C^{IMU}_{Cam} (I_3 - C^t_{T_{in}}) T^{IMU}_{Cam}
\end{aligned}
$$

Now we want to express this equation in the local reference frame:

$$
C^{IMU}_{Cam} P^i_{T_{in}}(T_{in}) = V(T_{in})\Delta_t + G\frac{\Delta_t^2}{2} + S_{T_{in}}(t) + C^{IMU}_{Cam} C^t_{T_{in}} P^i_t(t) - C^{IMU}_{Cam} (I_3 - C^t_{T_{in}}) T^{IMU}_{Cam} \quad (7.3)
$$

By using the same notation we introduced in Equation 3.4, we obtain the equation:

$$
S_j = C^{IMU}_{Cam} (\lambda^i_1 \mu^i_1 - \lambda^i_j \mu^i_j + (I_3 - C^{t_j}_{T_{in}}) T^{IMU}_{Cam}) - V t_j - G\frac{t_j^2}{2} \quad (7.4)
$$

Note that, if we have the knowledge of C^{IMU}_{Cam}, Equation 7.4 becomes a linear equation system in the same unknowns of 3.9 with the additional unknown T^{IMU}_{cam}. Note that a method to compute C^{IMU}_{Cam} independently is proposed in [15]. Hence, starting from 7.4 it is possible to obtain a new closed-form solution that also determines the translation T^{IMU}_{Cam}.

In our case we assume both the knowledge of C^{IMU}_{Cam} and T^{IMU}_{Cam}. Indeed, unless the MAV changes its geometry throughout an operation (such as a foldable drone) it is better to perform calibration offline in order to reduce the number of unknowns for the state estimation.

We have to modify the linear system accordingly to take this into account. In the matrix Ξ, μ^i_1 and μ^i_j are now multiplied by C^{IMU}_{Cam}.

The term $C^{IMU}_{Cam}(I_3 - C^{t_j}_{T_{in}}) T^{IMU}_{Cam}$ is fully determined and does not multiply any unknown, it is therefore added to the S vector.

The linear system therefore becomes:

$$
S \equiv [(S_2 - \Psi_2)^T, ..., (S_2 - \Psi_2)^T, (S_3 - \Psi_3)^T, ..., (S_3 - \Psi_3)^T, ..., (S_{n_i} - \Psi_{n_i})^T, ..., (S_{n_i} - \Psi_{n_i})^T]^T
$$

$$
\Xi \equiv \left[\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
T_2 & S_2 & \varphi^1_1 & 0_3 & 0_3 & -\varphi^1_2 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\
T_2 & S_2 & 0_3 & \varphi^2_1 & 0_3 & 0_3 & -\varphi^2_2 & 0_3 & 0_3 & 0_3 & 0_3 \\
... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
T_2 & S_2 & 0_3 & 0_3 & \varphi^N_1 & 0_3 & 0_3 & -\varphi^N_2 & 0_3 & 0_3 & 0_3 \\
... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
T_{n_i} & S_{n_i} & \varphi^1_1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\varphi^1_{n_i} & 0_3 & 0_3 \\
T_{n_i} & S_{n_i} & 0_3 & \varphi^2_1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\varphi^2_{n_i} & 0_3 \\
... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
T_{n_i} & S_{n_i} & 0_3 & 0_3 & \varphi^N_1 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & -\varphi^N_{n_i}
\end{array}\right]
$$

Where $\Psi_j = C^{IMU}_{Cam}(I_3 - C^{t_j}_{T_{in}}) T^{IMU}_{Cam}$ and $\varphi^i_j = C^{IMU}_{Cam} \mu^i_j$.

Feature extraction and matching

In this report, we considered that the camera provided bearing angles to observed point-features in the local frame. However, a standard monocular camera provides an array of pixels. In this section we explain how to extract those bearing angles over time from the sequence of images provided by the camera.

The first step is to find interest points in a provided image. These points have a rich local structure around them which depicts a well-defined position in 3D space. Therefore, it is possible to track these points over a time sequence from different point of view, when the camera is in motion.

We use an off-the-shelf computer vision algorithm to detect those points, the FAST corner detection described in [22] [23]. This technique, on top of being fast, is robust to very rapid translations, rotations and accelerations that occurs in MAV navigation.

Once these interest points are extracted we have to robustly match them with the previous interest points from the previous frame. The FAST corner detection also provides a method to perform the matching described in [22].

Bibliography

[1] Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y Ng, and Sebastian Thrun. Discriminative Training of Kalman Filters. *Proceedings of Robotics: Science and Systems I*, pages 289–296, 2005.

[2] L. Armesto, J. Tornero, and M. Vincze. Fast Ego-motion Estimation with Multi-rate Fusion of Inertial and Vision. *The International Journal of Robotics Research*, 26(6):577–589, 2007.

[3] Marco Bibuli, Massimo Caccia, and Lionel Lapierre. Sliding Window Filter with Application to Planetary Landing. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 7(PART 1):81–86, 2007.

[4] J. Dias, M. Vinzce, P. Corke, and J. Lobo. Editorial: Special Issue: 2nd Workshop on Integration of Vision and Inertial Sensors. *The International Journal of Robotics Research*, 26(6):515–517, 2007.

[5] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1456, 2013.

[6] Matthias Faessler, Flavio Fontana, Christian Forster, and Davide Scaramuzza. Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor. In *International Conference on Robotics & Automation*, 2015.

[7] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, 2008.

[8] P. Gemeiner, P. Einramhof, and M. Vincze. Simultaneous Motion and Structure Estimation by Fusion of Inertial and Vision Data. *The International Journal of Robotics Research*, 26(6):591–605, 2007.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] Richard I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[11] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. On the complexity and consistency of UKF-based SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4401–4408, 2009.

[12] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[13] Bo Li, Lionel Heng, Gim Hee Lee, and Marc Pollefeys. A 4-point algorithm for relative pose estimation of a calibrated camera with a known relative rotation angle. *IEEE International Conference on Intelligent Robots and Systems*, pages 1595–1601, 2013.

[14] Hongdong Li and Richard Hartley. Five-point motion estimation made easy. *Proceedings - International Conference on Pattern Recognition*, 1:630–633, 2006.

[15] M. Li and a. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[16] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections, 1981.

[17] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012.

[18] Agostino Martinelli. Closed-form solution of visual-inertial structure from motion. *International Journal of Computer Vision*, 106(2):138–152, 2014.

[19] Agostino Martinelli and Roland Siegwart. Vision and IMU Data Fusion: Closed-Form Determination of the Absolute Scale, Speed, and Attitude. *Handbook of Intelligent Vehicles*, 28(1):1335–1354, 2012.

[20] Najib Metni, Jean Michel Pflimlin, Tarek Hamel, and Philippe Souères. Attitude and gyro bias estimation for a flying UAV. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 295–301, 2005.

[21] D Nister. An efficient solution to the five point relative pose problem. pages 195–202, 2003.

[22] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. *Proceedings of the IEEE International Conference on Computer Vision*, II:1508–1515, 2005.

[23] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:430–443, 2006.

[24] Angelo Maria Sabatini. Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation. *Sensors*, 11(10):9182–9206, 2011.

[25] Vasiliy M. Tereshkov. An intuitive approach to inertial sensor bias estimation. *International Journal of Navigation and Observation*, pages 1–6, 2013.

[26] Stephan M Weiss. Vision Based Navigation for Micro Helicopters (PhD Thesis - Weiss 2012). (20305), 2012.

[27] Roni Yadlin. Attitude Determination and Bias Estimation Using Kalman Filtering. pages 1–11, 2008.