J Intell Robot Syst (2011) 61:287–299
DOI 10.1007/s10846-010-9490-z

Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM

Gabriel Nützi · Stephan Weiss · Davide Scaramuzza · Roland Siegwart

Received: 1 February 2010 / Accepted: 1 September 2010 / Published online: 12 November 2010

Abstract The fusion of inertial and visual data is widely used to improve an object's pose estimation. However, this type of fusion is rarely used to estimate further unknowns in the visual framework. In this paper we present and compare two different approaches to estimate the unknown scale parameter in a monocular SLAM framework. Directly linked to the scale is the estimation of the object's absolute velocity and position in 3D. The first approach is a spline fitting task adapted from Jung and Taylor and the second is an extended Kalman filter. Both methods have been simulated offline on arbitrary camera paths to analyze their behavior and the quality of the resulting scale estimation. We then embedded an online multi rate extended Kalman filter in the Parallel Tracking and Mapping (PTAM) algorithm of Klein and Murray together with an inertial sensor. In this inertial/monocular SLAM framework, we show a real time, robust and fast converging scale estimation. Our

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 231855 (sFly).
Gabriel Nützi is currently a Master student at the ETH Zurich. Stephan Weiss is currently PhD student at the ETH Zurich. Davide Scaramuzza is currently senior researcher and team leader at the ETH Zurich. Roland Siegwart is full professor at the ETH Zurich and head of the Autonomous Systems Lab.

G. Nützi · S. Weiss (✉) · D. Scaramuzza · R. Siegwart
ETH Autonomous Systems Laboratory, 8092, Zurich, Switzerland,
e-mail: stephan.weiss@mavt.ethz.ch
URL: www.asl.ethz.ch

G. Nützi
e-mail: nuetzi@student.ethz.ch

D. Scaramuzza
e-mail: davide.scaramuzza@ieee.org

R. Siegwart
e-mail: rsiegwart@ethz.ch

approach does not depend on known patterns in the vision part nor a complex temporal synchronization between the visual and inertial sensor.

Keywords IMU vision fusion · Absolute scale · Monocular SLAM · Kalman filter

1 Introduction

Online pose estimation with sensors on board is important for autonomous robots. We use an Inertial Measurement Unit (IMU), which is able to measure the 3D acceleration and rotation of a moving object and a fisheye camera. With each sensor it is possible to obtain the actual position of the moving vehicle. An integration of the acceleration measurements over time from the IMU yields a position in meters whereas an applied SLAM (Simultaneous Localization and Mapping) algorithm on the vision data provides a position with unknown scale factor λ. By using two cameras, which is not the case in this study, the scale problem would be solved. However, for example due to weight restrictions, often only one single camera can be applied. The estimate of the scale factor is essential to fuse the measurements of both camera and IMU. This fusion leads to a drift free estimation of the vehicles absolute position and velocity. Both are crucial parameters for efficient control.

We present two different methods for the scale estimation. The first is an online spline fitting approach adapted from Jung and Taylor [1]. The second is a multi rate Extended Kalman Filter (EKF).

The essence of this study, besides the scale estimation, was also to have a completely different approach at hand, the spline fitting, which we then can compare to the so often used EKF. Both approaches have been simulated in Matlab and compared to each other. The EKF has then been implemented online, because of its better performance. For the implementation, we modified the source code in the PTAM algorithm of Klein and Murray [2]. and included some additional parallel tasks which allowed us to filter both data with a relatively high sample rate. The novelty in this paper is the possibility to estimate the absolute scale in real time only with the help of a single camera and a 3D accelerometer. This simple, yet effective method is applicable (on board) on any vehicle featuring these two devices. Yielding also the absolute position and velocity of the vehicle the proposed algorithm is directly applicable for control.

This paper is organized as follows: In Section 2 we discuss the related work. Section 3 gives an overview of hardware and inputs used for the algorithms. In Sections 4 and 5 the two approaches are outlined and the results are provided for each of them. They are then discussed in the last section.

2 Related Work

Fusion of vision and IMU data can be classified into *Correction*, *Colligation* and *Fusion*. Whereas the first uses information from one sensor to correct or verify another, the second category merges different parts of the sensors. For example, Nygards [3] integrated visual information with GPS to correct the inertial system.

Labarosse [4] suggested using inertial data to verify the results from visual estimation. Zuffery and Floreano [5] developed a flying robot equipped only with a low resolution visual sensor and a MEMS rate gyro. They introduced gyro data into the absolute optical flow calculations of two cameras.

EKFs are widely used on the third category. In this paper we focus on the scale problem which arises due to the combination of monocular vision and inertial data. Unfortunately, most studies do not consider the scale ambiguity. For example Huster [6] implemented an EKF to estimate a relative position of an autonomous underwater vehicle, but the scale has not been included in the state vector. Arnesto et al. [7] fused orientation and position both from camera and inertial sensors in a multi rate EKF for a 6D tracking task. Stereo vision is used to solve the scale ambiguity directly. Stereo vision was further used in [7–10]. Like in [9], huge dimensional states render real time implementations difficult. Stratmann and Solda [11] used a KF to fuse gyro data with optical flow calculations. Because no translation was recovered, no scale factor had to be estimated. Also the authors of [12] and [13] only included orientation. In [14] they used a single camera to track lane boundaries on a street for autonomous driving. The authors did not provide information about the scale. Also in Eino et al. [15], where inertial data from an IMU is fused with the velocity estimation from a vision algorithm, no details about the scale problem are reported. Ribo et al. [16] proposed an EKF to fuse vision and inertial data to estimate the 6DoF attitude. There and in [10, 17–19] the authors use a priori knowledge to overcome the scale problem. Aid of an unscented Kalman filter, Kelly estimated the 6 degrees of freedom (DoF) between a camera and an IMU on a rigid body [20]. In his most recent work he did not use a known target but uses SLAM and estimates thus also the absolute scale in the offline estimation as the algorithm is too slow for real time implementation. In this study we present an online multi rate EKF which provides an accurate estimate of the scale factor λ in a converging time as fast as 15 s. The filter can be implemented online on any vehicle featuring a camera and a 3D accelerometer. The novelty of our proposed method does not lie in the sensor fusion as such, but more on its real-time implementation with the focus to apply it on an embedded platform, thanks to small matrices in the multi rate EKF. Our approach does neither rely on a complex temporal calibration of the sensors.

3 Setup and Inputs

For the vision input we used a USB uEye UI-122xLE camera from IDS with a fisheye lens. The camera has a resolution of 752 × 480 and a frame rate up to 87 fps. It's high dynamic range and global shutter minimized motion blur.

For the inertial inputs, we used the solid state vertical gyro VG400CC-200 from Crossbow with up to 75 Hz, which includes a tri-axial accelerometer and a tri-axial gyroscope. It has an input range of ±10 g with a resolution <1.25 mg. Internally, a Kalman filter is run which already corrects for the bias and misalignment. The IMU output is the rotation yaw, pitch and roll and the 3D acceleration a_x, a_y, a_z in its coordinate system **C**.

The uEye camera was mounted underneath the Crossbow IMU which is shown in Fig. 1a. The calibration to obtain the rotation matrix $\mathbf{R}_{ca}$ was calculated with the InerVis IMU CAM calibration toolbox in Matlab [21], see Fig. 1b. The subscript

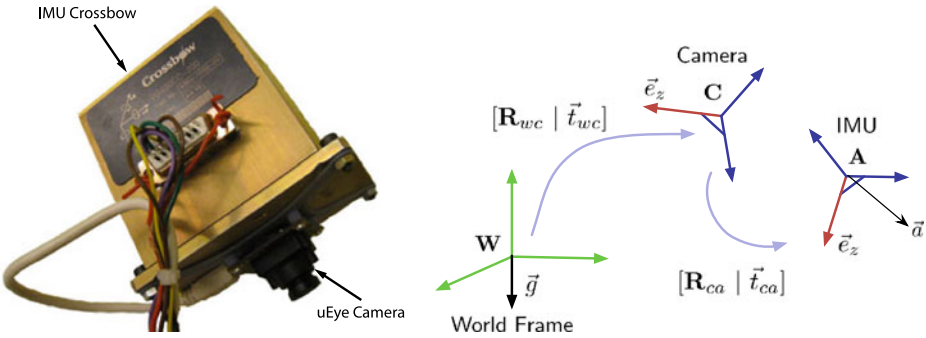


Fig. 1 **a** The camera/IMU setup. **b** Schematics of the reference frames. The subscripts 'wc' denote the rotation from the camera frame **C** to the world frame **W** and vice versa. The acceleration $\vec{a}$ is measured in the IMU frame and is then resolved in the world frame where we subtract the gravity vector $\vec{g}$

'ca' denotes the rotation from the acceleration frame **A** to the camera frame **C**. In Fig. 1b, there are three relevant reference frames, **W**, the fixed world frame, the camera reference frame **C** and the IMU frame **A** which are mounted together.

4 Spline Fitting

4.1 Overview

Jung and Taylor considered the problem of estimating the trajectory of a moving camera by combining the measurements obtained from an IMU with the results obtained from a structure from motion (SFM) algorithm. They proposed an offline spline fitting method, where they fit a second order spline into a set of several keyframes obtained by the SFM algorithm with monocular vision. We used this approach because of the good results reported by the authors. We modified this spline fitting so that we can implement it for an online scale estimation. Figure 2 shows this concept. The black line belongs to the ground truth path of the Camera/IMU. The blue points are the camera poses obtained from the visual SLAM algorithm. We assumed noise with a standard deviation of σ_v varying from 0.01 to 0.05 m/λ, which is close to reality for the chosen SLAM algorithm. For this visualisation we assume a scale factor $\lambda = 1$. We divide our time line into sections with a fixed duration of $T_s = 0.5$ to 1.5 s. For each section, three to five second order splines are fitted (green dashed line). The red curve shows the integrated IMU over the time T_s which is flawed by its drift. The spline fitting is done continuously when a new section is finished. This introduced time delay of 0.5 to 1.5 s. Note that this can be critical if used in control, albeit the scale is not prone to sudden changes. By introducing a weighting into the least square optimization, it is possible to rely more on the acceleration measurements when the quality of the vision tracking deteriorates as

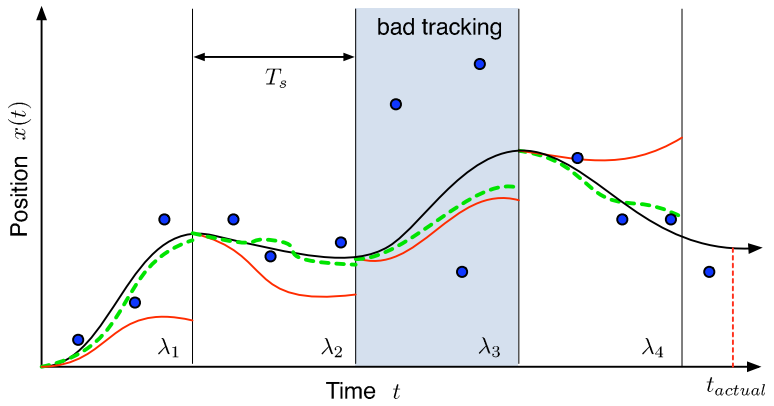


Fig. 2 Concept for the online spline fitting, only $x(t)$ is visualized. In each section T_s we fit 3–5 s order splines *(green dashed line)* into the *blue points*, which are the position without scale from the SLAM algorithm. The *red line* shows the position obtained from integration of the acceleration data over time. Each section has its own scale λ

shown in Fig. 2 'bad tracking'. For each section we estimate one scale factor λ_n, which is then averaged over time.

For the simulation in Matlab on arbitrary camera paths, we simplified the following aspects compared to the original version from Jung and Taylor:

- We assumed that the acceleration of the CAM/IMU is already given in the world frame. Therefore the subtraction of the gravitation vector $\vec{g}$ is not necessary.
- The keyframe position are now soft constraints, meaning that a least square is done between the acceleration measurements and the keyframe points. This differs from Jung and Taylor, where the keyframes are hard constraints for the fitted spline curve. This allows more noise and even short periods of wrong tracking in the visual part.

4.2 Least Square Optimisation

We obtain the following equations for the least square in one section T_s:

The second order spline yields,

$$\min \sum_{i=1}^{3 \text{ to } 5} \left\| \begin{pmatrix} a_i^x + b_i^x \tau_j + c_i^x \tau_j^2 - \lambda \hat{x}_j \\ a_i^y + b_i^y \tau_j + c_i^y \tau_j^2 - \lambda \hat{y}_j \\ a_i^z + b_i^z \tau_j + c_i^z \tau_j^2 - \lambda \hat{z}_j \end{pmatrix} \right\|^2 \tag{1}$$

Where $[a_i, b_i, c_i]$ for each direction $[x, y, z]$ and λ are the unknown parameters. The subscript 'i' denotes the spline number running from 3 up to 5. $\tau_j = (t_j - iT_s)/T_s$ and $[\hat{x}_j, \hat{y}_j, \hat{z}_j]$ are the positions from the SLAM algorithm spaced over one section (blue points in Fig. 2).

The continuity constraints at the boundaries of the splines in each section are,

$$\begin{pmatrix} a_{i+1}^x \\ a_{i+1}^y \\ a_{i+1}^z \end{pmatrix} = \begin{pmatrix} a_i^x + b_i^x + c_i^x \\ a_i^y + b_i^y + c_i^y \\ a_i^z + b_i^z + c_i^z \end{pmatrix} \tag{2}$$

$$\begin{pmatrix} b_{i+1}^x \\ b_{i+1}^y \\ b_{i+1}^z \end{pmatrix} = \begin{pmatrix} b_i^x + 2c_i^x \\ b_i^y + 2c_i^y \\ b_i^z + 2c_i^z \end{pmatrix} \tag{3}$$

The acceleration from spline yields:

$$\vec{a}_w = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \frac{2}{T_s^2} \begin{pmatrix} c_i^x \\ c_i^y \\ c_i^z \end{pmatrix} \tag{4}$$

$$\min \sum_{k=1}^{m} \|\hat{\vec{a}}_w(t_k) - \vec{a}_w(t_k)\|^2 \tag{5}$$

Where the subscript 'w' denotes the acceleration resolved in the world frame **W** and $\hat{\vec{a}}$ are the noisy acceleration measurements from the IMU.

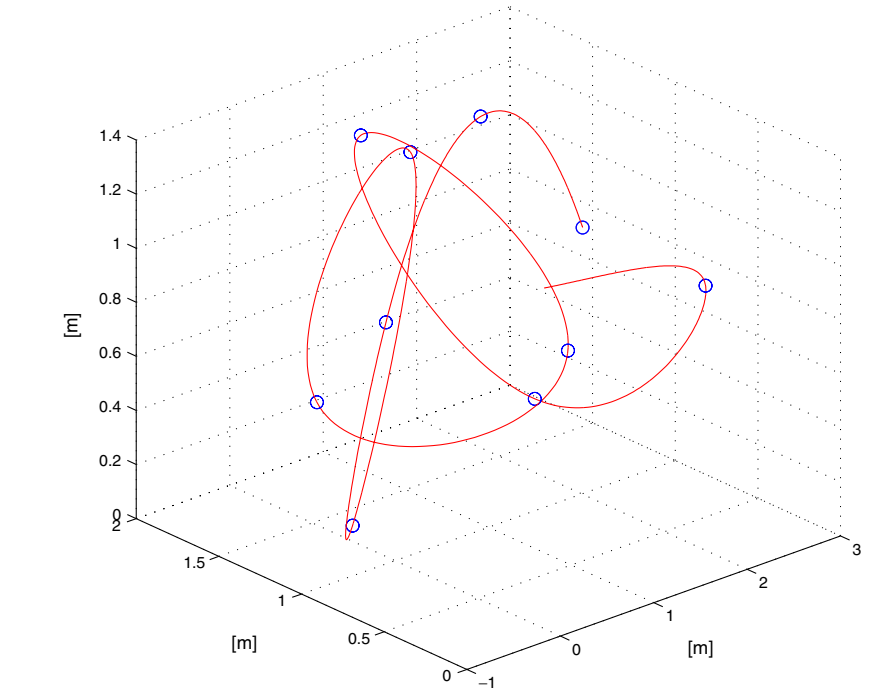
With Eqs. 1 and 5 we can form a least square problem in the form $\mathbf{A}\vec{z} = \vec{b}$ with constraints $\mathbf{C}\vec{z} = 0$ from Eqs. 2 and 3. The analytical solution to this problem can be found with Lagrange multipliers. We solved this in Matlab with the function `linsq`, which provides linear constraints in a least square problem.

4.3 Results

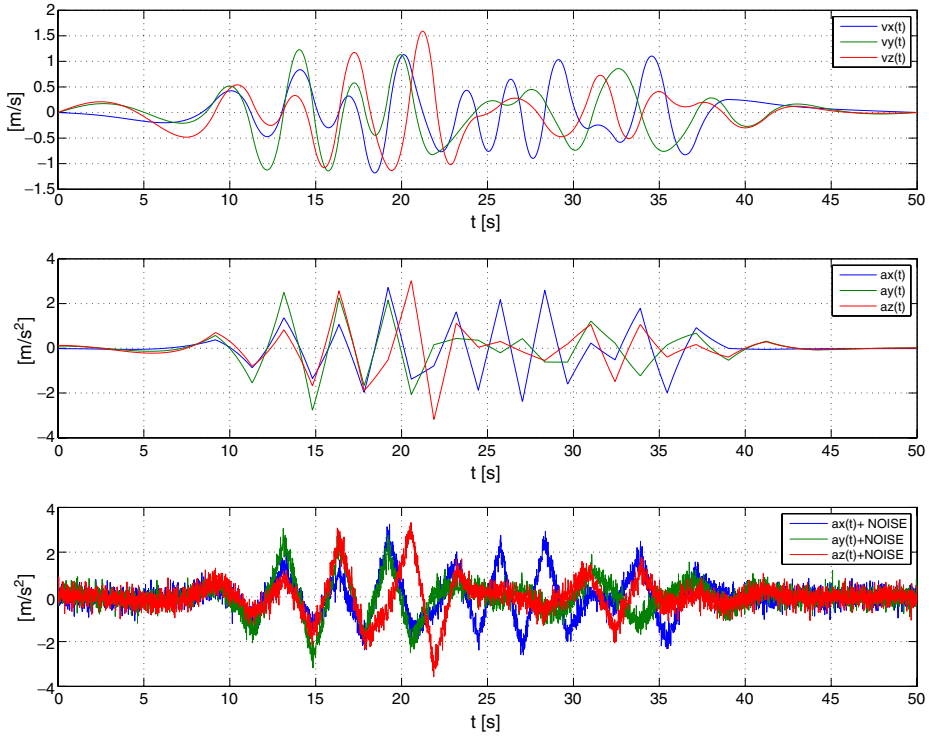
In Matlab, we simulated a 3D camera path in a 2 m cube, shown in Fig. 3a with its appropriate velocity and acceleration curves shown in Fig. 3b. For the acceleration measurements, noise with standard deviation varying from 0.1–0.5 m/s^2 was assumed.

Despite the fact that Jung and Taylor achieved good results with his spline fitting task, our scale estimation was not promising. Even with our simpler adaption which should be more robust, the resulting estimation for the scale λ was quite random and was heavily dependent on the noise which was added to the real acceleration and also on the shape of the 3D camera path. The resulting scale estimates were from superb (error of 5% and less) to very bad (error of 50% and more) with different noise and different camera paths. The reason which accounts for this poor estimation is mainly due to the ill conditioned matrix **A**, which should be preconditioned beforehand and also to the fact that almost only the spline variables c_i are in the matrix **A** which results in bad estimation of all the other parameters. The online spline fitting approach could not be realized satisfyingly, because the performance of a normal spline fitting with 3 to 5 splines in a section T_s was not accurate at all. It turned out

Fig. 3 Simulated path in its spacial and temporal representation: **a** The simulated 3D camera path ▶ in Matlab. **b** The simulated velocity, acceleration and the acceleration with added noise from the 3D spline in Matlab


[m]
[m]
[m]

(a)


vx(t)
vy(t)
vz(t)
[m/s]
t [s]
ax(t)
ay(t)
az(t)
[m/s²]
t [s]
ax(t)+ NOISE
ay(t)+NOISE
az(t)+NOISE
[m/s²]
t [s]

(b)

that the spline fitting task becomes more and more accurate the longer our time line is. With up to 100 splines in 10 s we achieved scale estimation errors of around 5% and less. This speaks for the suggested offline implementation by Jung and Taylor. On the other hand, a duration of $T_s = 10$ s introduces a far too long lag into the system, what makes an online application for a controller rather critical.

5 Extended Kalman Filter

5.1 EKF State Representation

Our final non-linear prediction model is ('k' denotes the time step),

$$\vec{z}_{k+1} = \vec{f}_k(\vec{z}_k) + \nu_k \tag{6}$$

$$\begin{pmatrix} \vec{x}_{k+1} \\ \vec{v}_{k+1} \\ \vec{a}_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{bmatrix} \mathbf{I}_3 & \frac{T}{\lambda}\mathbf{I}_3 & \frac{T^2}{2\lambda}\mathbf{I}_3 & 0 \\ 0 & \mathbf{I}_3 & T\mathbf{I}_3 & 0 \\ 0 & 0 & \mathbf{I}_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \vec{x}_k \\ \vec{v}_k \\ \vec{a}_k \\ \lambda_k \end{pmatrix} \tag{7}$$

where $\vec{x}_{k+1} \in \mathbb{R}^{3\times 1}$ is the position without scale of the IMU/Camera and $\vec{v}_{k+1}, \vec{a}_{k+1} \in \mathbb{R}^{3\times 1}$ are the velocity and acceleration of the IMU/Camera in metric unit $[m]$. ν_k is the gaussian process noise. Equation 7 is a simple discrete integration over the varying time T. Every vector in $\vec{z}_k$ is resolved in the world frame $\mathbf{W}$. Note that we do not include the orientation information in the model nor use it as a measurement in order to keep the algorithm simple and fast. On each acceleration measurement we do the conversion from the inertial to the world frame by using a zero order hold (ZOH) of the unfiltered attitude measurement returned by the visual SLAM framework. As we work in a middle size environment with enough loops we assume negligible drift in the SLAM map and assume thus highly accurate attitude estimation from the visual SLAM framework. The model in its linearised form yields,

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{I}_3 & \frac{T}{\lambda}\mathbf{I}_3 & \frac{T^2}{2\lambda}\mathbf{I}_3 & -\frac{T}{\lambda^2}\mathbf{I}_3 - \frac{T^2}{2\lambda^2}\mathbf{I}_3 \\ 0 & \mathbf{I}_3 & T\mathbf{I}_3 & 0 \\ 0 & 0 & \mathbf{I}_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

To implement the fusion we consider the measurements in different observation vectors. For a multirate filter, as it is in our case, the literature suggests two solutions. One would be using a (higher order) hold to synchronize the different measurements. Another is to weight the uncertainty of the measurement according to its temporal occurrence. We used the simplification in [12]. We claim no certainty at all if no measurement is available (i.e. the measurement noise variance is infinite). Thus the update equations simplify to Eqs. 11–13 if vision data arrive or to Eqs. 14–16 if IMU data arrive. A more complex weighting function (i.e. exponential decay in time) could also be applied, however, at the cost of speed.

The measurement updates for the vision and the IMU yields ('V' and 'I' denotes Vision and IMU)

$$\vec{y}_{V,k} = \mathbf{H}_{V,k}\vec{z}_k = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & 0\end{bmatrix}\vec{z}_k \tag{9}$$

$$\vec{y}_{I,k} = \mathbf{H}_{I,k}\vec{z}_k = \begin{bmatrix}\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & 0\end{bmatrix}\vec{z}_k \tag{10}$$

The innovation for the vision part is,

$$\mathbf{K}_{V,k} = \mathbf{P}_k^- \mathbf{H}_{V,k}^\top \left(\mathbf{H}_{V,k}\mathbf{P}_k^- \mathbf{H}_{V,k}^\top + \mathbf{R}_V\right)^{-1} \tag{11}$$

$$\vec{z}_k = \vec{z}_k + \mathbf{K}_{V,k}\left(\vec{x}_{\text{SLAM}} - \mathbf{H}_{V,k}\vec{z}_k\right) \tag{12}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_{V,k}\mathbf{H}_{V,k}\right)\mathbf{P}_k^- \tag{13}$$

The innovation for the IMU part is,

$$\mathbf{K}_{I,k} = \mathbf{P}_k^- \mathbf{H}_{I,k}^\top \left(\mathbf{H}_{I,k}\mathbf{P}_k^- \mathbf{H}_{I,k}^\top + \mathbf{R}_I\right)^{-1} \tag{14}$$

$$\vec{z}_k = \vec{z}_k + \mathbf{K}_{I,k}\left(\vec{a}_{\text{IMU}} - \mathbf{H}_{I,k}\vec{z}_k\right) \tag{15}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_{I,k}\mathbf{H}_{I,k}\right)\mathbf{P}_k^- \tag{16}$$

The two matrices $\mathbf{R}_I, \mathbf{R}_V$ are the noise covariance matrices for the vision and IMU measurement inputs $\vec{x}_{\text{SLAM}}, \vec{a}_{\text{IMU}}$ which are resolved in the world frame $\mathbf{W}$. The vector $\vec{x}_{\text{SLAM}}$ is the position without scale obtained from the vision algorithm (SLAM). The IMU measurement $\vec{a}_{\text{IMU}}$ needs special attention, because significant errors arise in the conversion from the raw IMU output. According to Fig. 1b, the acceleration in the world frame is given by,

$${}_w\vec{a} = \mathbf{R}_{wc}\mathbf{R}_{ca}\left({}_a\vec{a} - \vec{b}\right) - {}_w\vec{g} \tag{17}$$

where ${}_a\vec{a}$ is the acceleration output from the Crossbow IMU in its coordinate frame $\mathbf{A}$. $\vec{b}$ is the static offset of the IMU. The rotation matrix $\mathbf{R}_{wc}$ is provided by the SLAM algorithm. This rotation is much more accurate than the rotation solely from the IMU which is integrated over time and suffers from drift. The accuracy of $\mathbf{R}_{wc}$ depends on the initialization of the map by the SLAM algorithm. With good initialization the error of the angles are around ± 1–$2°$. This error is due to the fact that we did not adapt the camera model in the SLAM algorithm of Klein and Murray, which is still perspective and not for fisheye lenses. Another issue which causes significant errors in the acceleration measurement ${}_c\vec{a}$ is the fact that our accelerations induced by the camera motion are small compared to the gravitational field. The subtraction of the gravitation vector ${}_w\vec{g}$ then introduces a dynamical bias in ${}_c\vec{a}$ when the camera rotation is inaccurate.

5.2 Results with Simulated and Real Data

We have simulated the proposed Kalman filter offline with our 3D path generator (see spline fitting) and also on real measurement sets. For each measurement set we

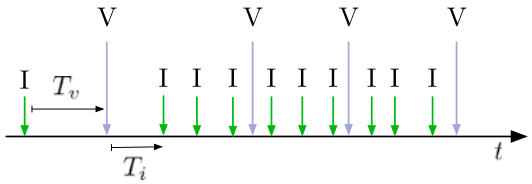


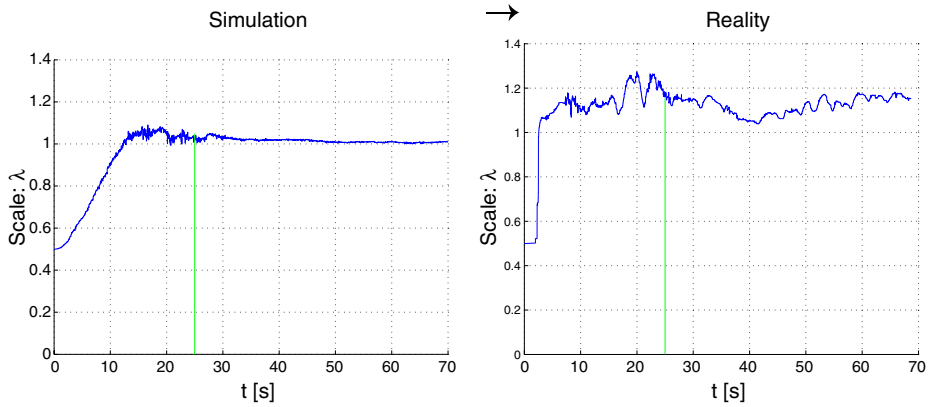
Fig. 4 Distribution of the measurements over time, (*I*: IMU measurement arrived, *V*: vision measurement arrived)

recorded the two measurement inputs for the Kalman filter, the vision non-scaled position from Klein and Murray's SLAM algorithm and the acceleration from the Crossbow IMU. For the measurement sets and for the path generator we used the Kalman filter in Eq. 7 but with three different setups. The first is the same as in Eq. 7. In the second we only use the Z-axis, meaning that the new state yields $[\vec{x}_z, \vec{v}_z, \vec{a}_z, \lambda]$ and in the third we use only the X and Y-axis. Figure 5 shows the scale estimation $\lambda(t)$ for the simulated 3D path on the left and for the measurement set on the right. Always the same path and measurement set was used. To make the measurements of the simulated 3D path close to the real data, we chose the same standard deviations for the acceleration noise of the simulated data as measured on the real data ($\sigma_{\text{SLAM}} = 0.01$ (with a $\lambda = 1 \rightarrow 0.01$ m), $\sigma_{\text{IMU}} = 0.2$ m/s^2) The initial velocity and acceleration for the state vector were set to zero. The initial scale λ was set to the worst case, where the initial value would have an error of 50%, which does not normally happen. A simple integration over time of the acceleration gives an initial guess for the scale with an error in a range of 5–20%. After 25 s we also lower the variance of λ in the state space noise covariance matrix $\mathbf{Q}$. This produces less overshoot and faster convergence. The two different correction updates, Eqs. 11 and 14, are performed as soon as the measurements arrive. The distribution over time looks as shown in Fig. 4. Before the correction we do a prediction either with T_i or T_v in the matrix $\mathbf{F}_k$.

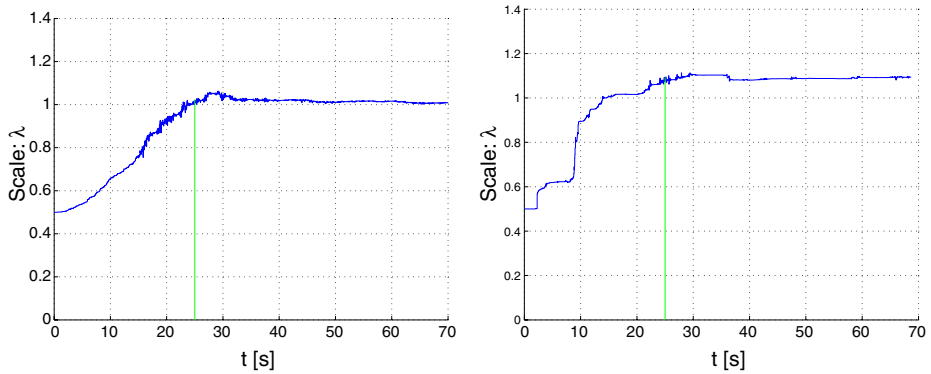
The simulation on the 3D path produced very nice results in any setup which is not surprising because we simulated the acceleration from the spline ideally and already resolved in the world frame, so Eq. 17 was not needed. This is also the reason why the plots in Fig. 5a and Fig. 5c do not differ from each other very much. The contrary shows the simulation on the real data. The fewer directions (X,Y,Z) are included in the Kalman filter, the more accurate the estimate becomes. The reason for this is mainly due to the influence of the measurement inputs on λ. Because we have significant errors which arise due to the conversion in Eq. 17, the acceleration in the world frame has a dynamic bias which is difficult to estimate. These wrong measurements influence λ (Kalman gain $\mathbf{K}$) and make the scale estimation very sensitive, which can be seen in Fig. 5b. Therefore, a one-axis-only estimate gives the best result. In our case, the Z-estimate proved to be much better than the X and Y, because in our Cam/IMU movements, the subtraction of the gravity vector gives the smallest errors on the Z acceleration.

5.3 Online Implementation Results

For an online implementation, we embedded only the third setup into the code of Klein and Murray [22]. The code is written in C++ and uses the computer vision



(a) X,Y,Z estimate with the 3D path (b) X,Y,Z estimate with real data



(c) Z estimate with the 3D path (d) Z estimate with real data

Fig. 5 On the *left*: Offline simulation of the Kalman filter with the simulated 3D path. On the *right*: Offline Simulation of the Kalman filter with real data. The *green line* shows the time when the Q Matrix is changed. The scale is fixed to $\lambda = 1$ for the 3D path and an exactly measured $\lambda = 1.07$ for the real data. The fewer directions (X,Y,Z) are included in the Kalman filter, the more accurate the estimate becomes. The reason for this is mainly due to the influence of the measurement inputs on λ. The Z-estimate proved to be much better than the X and Y, because in our Cam/IMU movements, the subtraction of the gravity vector gives the smallest errors on the Z acceleration

library `libcvd` and the numeric library `TooN`. The work flow of our implementation is shown in Fig. 6. The original code consists of 2 threads, the Tracker and the MapMaker. The core thread, the Tracker, is responsible for the tracking of the incoming video frames and provides a translation $\vec{t}_{wc}$ and a rotation matrix $\mathbf{R}_{wc}$ of the camera. The MapMaker is responsible for the storage of the keyframes and executes the bundle adjustment over the whole map. The MapMaker also pokes the recovery algorithm when the Tracker gets lost.

We added the IMU thread and the Kalman thread. The IMU thread provides the acceleration measurements from the Crossbow IMU. The Kalman thread starts with the initial position from the SLAM algorithm and with velocity and acceleration set

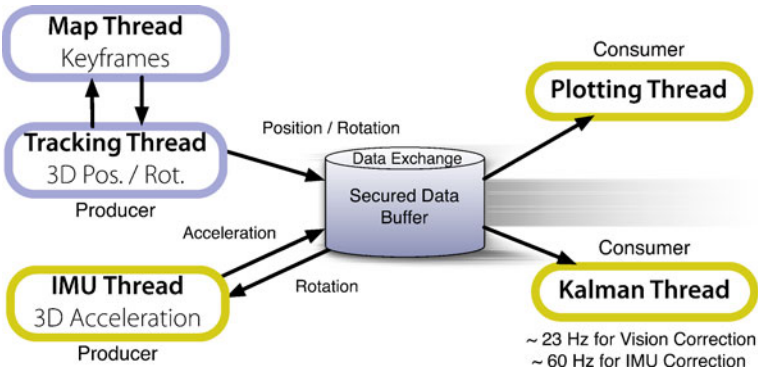


Fig. 6 Work flow of the online implementation. The original SLAM algorithm consists of the Tracking and the MapMaker thread *(blue)*. We added two more threads *(yellow)* for the scale estimation. The IMU thread which provides the IMU acceleration data and the Kalman filter thread itself which estimates the scale factor λ

to zero. The value for the initial λ is calculated by integration of the acceleration over 1 s beforehand.

We also introduced time-varying values into the covariance matrix **Q**, which allows a certain control of the sensitivity of the Kalman filter. Additionally, we suspend the Kalman filter thread when the Tracker is lost. After the map has been recovered we continue the Kalman filter, but with newly acquired values for the state space. The velocity is set to zero, because we do not have an accurate velocity measurement at hand. The error covariance matrix **P** is not reset.

6 Conclusion and Future Work

This paper describes two different approaches to estimate the absolute scale factor of a monocular SLAM framework. We analyzed and compared a modified spline fitting method by Jung and Taylor to the EKF. We revealed the limitations of the spline method with respect to an online implementation. We implemented then a multi rate EKF running in real time with over 60 Hz. The filter produced good and accurate results within a very fast converging time of down to 15 s. Our approach is kept simple yet effective in order to be applied on any vehicle featuring a camera and a 3D accelerometer. We are not dependent on any known pattern in the visual framework, nor on a complex temporal calibration of camera and IMU. Using the estimated scale factor the vehicle can be controlled on its absolute position and velocity.

References

1. Jung, S.-H., Taylor, C.: Camera trajectory estimation using inertial sensor measurements and structure from motion results. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. II–732–II–737 (2001)
2. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan (2007)

3. Nygårds, J., Skoglar, P., Ulvklo, M., Högström, T.: Navigation aided image processing in uav surveillance: preliminary results and design of an airborne experimental system. J. Robot. Syst. **21**(2), 63–72 (2004)
4. Labrosse F.: The visual compass: performance and limitations of an appearance-based method. JFR **23**(10), 913–941 (2006)
5. Zufferey, J.-C., Floreano, D.: Fly-inspired visual steering of an ultralight indoor aircraft. IEEE Trans. Robot. **22**(1), 137–146 (2006)
6. Huster, A., Frew, E., Rock, S.: Relative position estimation for auvs by fusing bearing and inertial rate sensor measurements. In: Oceans '02 MTS/IEEE, vol. 3, pp. 1863–1870 (2002)
7. Armesto, L., Chroust, S., Vincze, M., Tornero, J.: Multi-rate fusion with vision and inertial sensors. In: 2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04, vol. 1, pp. 193–199, 1 April–1 May 2004
8. Kim, S.-B., Lee, S.-Y., Choi, J.-H., Choi, K.-H., Jang, B.-T.: A bimodal approach for gps and imu integration for land vehicle applications. In: 2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall, vol. 4, pp. 2750–2753 (2003)
9. Chroust, S.G., Vincze, M.: Fusion of vision and inertial data for motion and structure estimation. J. Robot. Syst. **21**(2), 73–83 (2004)
10. Helmick, D., Cheng, Y., Clouse, D., Matthies, L., Roumeliotis, S.: Path following using visual odometry for a mars rover in high-slip environments. In: 2004 IEEE Aerospace Conference, 2004. Proceedings, vol. 2, pp. 772–789 (2004)
11. Stratmann, I., Solda, E.: Omnidirectional vision and inertial clues for robot navigation. J. Robot. Syst. **21**(1), 33–39 (2004)
12. Niwa, S., Masuda, T., Sezaki, Y.: Kalman filter with time-variable gain for a multisensor fusion system. In: 1999 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI '99. Proceedings, pp. 56–61 (1999)
13. Waldmann J.: Line-of-sight rate estimation and linearizing control of an imaging seeker in a tactical missile guided by proportional navigation. IEEE Trans. Control Syst. Technol. **10**(4), 556–567 (2002)
14. Goldbeck, J., Huertgen, B., Ernst, S., Kelch, L.: Lane following combining vision and dgps. Image Vis. Comput. **18**(5), 425–433(9) (2000)
15. Eino, J., Araki, M., Takiguchi, J., Hashizume, T.: Development of a forward-hemispherical vision sensor for acquisition of a panoramic integration map. In: IEEE International Conference on Robotics and Biomimetics, 2004. ROBIO 2004, pp. 76–81 (2004)
16. Ribo, M., Brandner, M., Pinz, A.: A flexible software architecture for hybrid tracking. J. Robot. Syst. **21**(2), 53–62 (2004)
17. Zaoui, M., Wormell, D., Altshuler, Y., Foxlin, E, McIntyre, J.: A 6 d.o.f. opto-inertial tracker for virtual reality experiments in microgravity. Acta Astronaut. **49**, 451–462 (2001)
18. Helmick, D., Roumeliotis, S., McHenry, M., Matthies, L.: Multi-sensor, high speed autonomous stair climbing. In: IEEE/RSJ International Conference on Intelligent Robots and System, 2002, vol. 1, pp. 733–742 (2002)
19. Robert Clark, R., Lin, M.H., Taylor, C.J.: 3d environment capture from monocular video and inertial data. In: Proceedings of SPIE, The International Society for Optical Engineering (2006)
20. Kelly, J., Sukhatme, G.: Fast relative pose calibration for visual and inertial sensors. In: Khatib, O., Kumar, V., Pappas, G. (eds.) Experimental Robotics, vol. 54, pp. 515–524. Springer Berlin/Heidelberg (2009)
21. Lobo, J.: InerVis IMU Camera Calibration Toolbox for Matlab. http://www2.deec.uc.pt/~jlobo/InerVis_WebIndex/InerVis_Toolbox.html (2008)
22. Klein, G.: Source Code of PTAM (Parallel Tracking and Mapping). http://www.robots.ox.ac.uk/~gk/PTAM/