Fusion of IMU and Vision for Absolute Scale Estimation

Gabriel Nützi, Davide Scaramuzza, Stephan Weiss, Roland Siegwart
Autonomous Systems Lab ETH Zürich

Abstract*— The fusion of inertial and vision data is a corner stone for obtaining a better pose estimation. In this paper we are looking at a Camera/IMU setup with one camera and a 3 axis accelerometer. We present and compare 2 different approaches to fuse inertial data with vision in order to estimate the unknown scale factor in an monocular SLAM framework. The first approach is a spline fitting task adapted from S. H. Jung and the second is an extended Kalman filter. Both methods have been simulated offline on arbitrary camera paths to analyse their behaviour and the quality of the resulting scale estimation. We then embedded an online extended Kalman filter thread and some other simple correction tasks in the Parallel Tracking and Mapping (PTAM) algorithm of Georg Klein. The extended Kalman filter has then been tested in real time with the IMU/Camera setup.

I. Introduction

Online pose estimation with sensors on board is important for autonomous robots, which is the basic foundation for making their decisions. In the view of bionics, cameras serve as eyes and inertial system as vestibule. The combination of vestibule and eye gives animals much more sensing flexibility to determine orientation in daily activities. The integration of inertial and visual sensors has a strong bionic foundation. For mobile robotics, (ground, air and underwater), localization of their position and motion detection is critically important.

In this paper we look at two sensors on board of an autonomous helicopter. The first is an Inertial Measurement Unit (IMU), which is able to measure the 3D acceleration and rotation of a moving object and the second is 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 (Simultanious Localisation and Mapping) algorithm on the vision data (video frames) provides a position with unknown scale factor λ. By using two cameras, which is not the case in this study, the scale ambiguity would be solved. Due to the lack of space, we cannot use binocular vision. By fusing these two positions, from vision and IMU, we obtain a better pose estimation of the vehicle, which is essential for stabilization and control of a micro helicopter. An accurate estimation of the scale factor λ is the main key to achieving this.

We present two different methods for the scale estimation. The first is an online spline fitting approach adapted from S.H Jung. [1]. The second is an extended Kalman filter.

Nowadays, the Kalman filter is probably the most commonly used algorithm for state estimation in linear and non-linear systems. 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 extended Kalman filter, which suffers from a huge performance loss compared to the standard linear Kalman filter. The normal linear Kalman filter produces optimal estimates by minimizing the variance of the estimation error.

Both approaches have been simulated in Matlab and compared to each other. The extended Kalman filter has then been implemented online, because of its better performance. Therefore, we modified the source code in the PTAM algorithm of Georg Klein [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 monocular camera and a tri-axial accelerometer.

This paper is organised as follows: In Section 2 we discuss the related work. Section 3 gives an overview of hardware which was used and the related input measurements for the fusion algorithm. In section 4 and 5 the two approaches are outlined and the results are provided for each of them, which are then discussed in the last section.

II. Related Work

Fusion of vision and IMU can be classified into 3 different categories. The first section is named *Correction*, where we use the results from one kind of sensor to correct or verify the data from another sensor. For example, Nygards [3] integrated visual information with GPS to correct the inertial system. Labarosse [4] suggested using interial data to verify the results from visual estimation.

The second category is *Colligation*, where one uses some variables resulting from the inertial data together with variables from the visual data. For example, Zuffery & 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. Another example is found in

[6] and [7] where the velocity was directly introduced into the visual measurement.

The third category is called *Fusion* and is by far the most popular method to efficiently combine inertial and visual data to improve pose estimation. The extended Kalman filter (EKF) also belongs to this category. EKF's are used on the fusion of inertial and vision data in [8–20].Extended Kalman filters are very powerful but suffer from the different sampling problem, which yields time-varying dynamic matrices, and also from the huge performance loss compared to the linear Kalman filter. In this paper we focus on the scale ambiguity which arises due to the combination of monocular vision and inertial data. Most studies do not consider the scale factor problem and assume it as given or calculate it through the known size of features from the vision data or they simply use binocular vision. For example Huster [21] implemented an extended Kalman filter to estimate a relative position to a feature of an autonomous underwater vehicle (AUV), but they did not estimate the scale factor with the Kalman filter. In Armesto et al.[8] they used an unscented Kalman filter with a camera stereo rig to circumvent the scale problem. In this study we present a Kalman filter which can be implemented online on a autonomous flying helicopter which provides an accurate estimate of the scale factor λ in 15-30 seconds, depending on the trajectory.

III. Setup & Inputs

A. Vision Input

We used a USB uEye UI-122xLE camera from IDS with a fish-eye lens. The camera has a resolution of 752 $\times$ 480 and a frame rate up to 87 fps. Due to its very low exposure time of 80 μs and to its high dynamic range, this camera produces very little motion blur in our camera movements, where the maximum velocity was below 5 m/s.

B. Inertial Input

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 $\pm$ 10 g with a resolution $<$ 1.25 mg. Internally, a Kalman filter is run which already corrects for the bias and missalignment. 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**.

C. Camera/IMU Setup

The uEye camera was mounted underneath the Crossbow IMU which is shown in Fig. 1. The calibration to obtain the rotation matrix $\mathbf{R}_{ca}$ was calculated with the InerVis IMU CAM calibration toolbox in Matlab [22], see Fig. 2. The subscript 'ca' denotes the rotation from the acceleration frame 'a' to the camera frame 'c'. In Fig. 2, there are 3 relevant reference frames, **W**, the fixed world frame, the camera reference frame **C** and the IMU frame **A** which are mounted together.

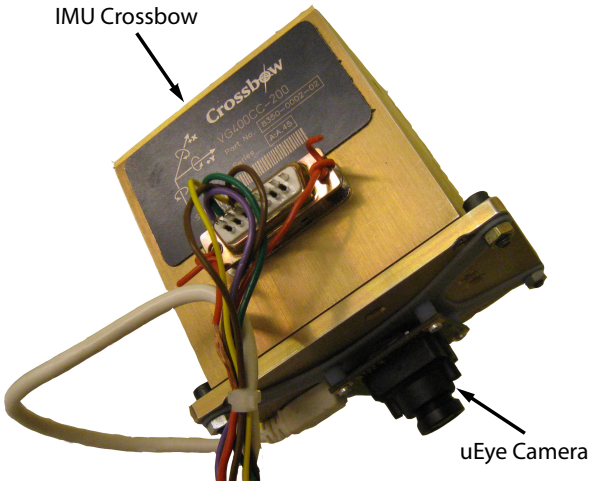


Fig. 1: The Camera/IMU setup

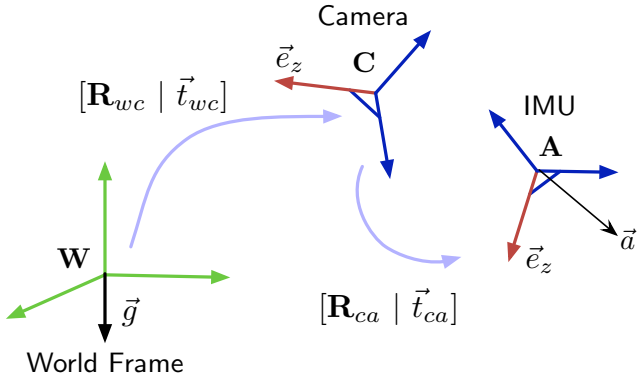


Fig. 2: Schematics of the reference frames. The subscripts 'wc' denote the rotation from the camera frame **C** to the world frame **W** and vice versa. The acceleration $\vec{a}$ is measured in the IMU frame and is then resolved in the world frame where we subtract the gravitiy vector $\vec{g}$.

IV. Spline Fitting

A. Overview

As already mentioned, we adapted the approach of Jung [1] for our situation. Jung 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 a spline fitting method, where they fit a second order spline into a set of several keyframes obtained by the SFM algorithm with monocular vision. They formulated this as an offline task. Our idea was to modify this spline fitting so that we can implement it for an online scale estimation. Fig. 3 shows this concept. The black line belongs to the ground truth path of the Camera/IMU. The blue points are the camera poses from the V-SLAM algorithm at the time the images were captured. Normally they are scaled by the unknown scale factor λ, but for this visualisation we assume a scale factor $\lambda = 1$. They are not located on the black curve because we assumed noise

with a standard deviation of σ_v varying from $0.01m/\lambda$ to $0.05m/\lambda$, which is close to reality for our SLAM algorithm. We devide our timeline into sections with a fixed duration $T_s = 0.5 - 1.5$ seconds. 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 lag of 0.5-1.5 seconds, which is still realizable, needs to be included in further control design , which could be critical for stability. By introducing a weighting into the least square optimisation, it is possible to rely more on the acceleration measurements when the quality of the vision tracking deteriorates as shown in Fig. 3 '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 fallowing aspects compared to the original version from Jung:

- → 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 performed.
- → 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, where the keyframes are hard constraints for the fitted spline curve (green dashed line). We changed this because Jung's version did not give satisfactory results in our case.

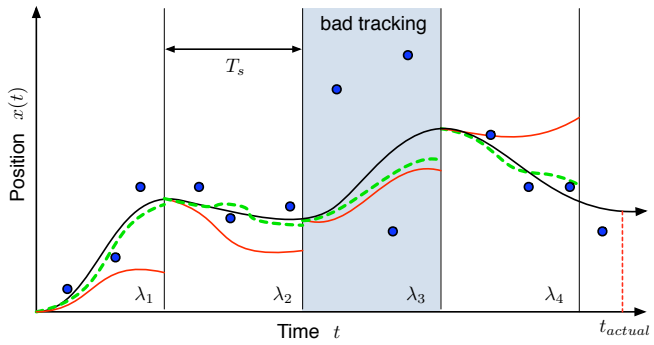


Fig. 3: Concept for the online spline fitting, only x(t) is visualized. In each section T_s we fit 3-5 second order splines (green dashed line) into the blue points, which are the position without scale from the SLAM algorithm. The red line shows the position obtained from integration of the acceleration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

B. Least Square Optimisation

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

The second order spline yields,

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

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. 3).

The continouity 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 $\mathbf{W}$ and $\hat{\vec{a}}$ are the noisy acceleration measurements from the IMU.

With equation 1 and 5 we can form a least square problem in the normal form $\mathbf{A}\vec{z} = \vec{b}$ with constraints $\mathbf{C}\vec{z} = 0$ from 2,3. 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.

C. Simulation

In Matlab, we simulated a 3D camera path in a 2 m cube, shown in Fig. 4 with its appropriate velocity and acceleration curves shown in Fig. 5.

For the acceleration measurements, noise with standard deviation varying from 0.1 to 0.5 m/s^2 was assumed.

D. Results

Despite the fact that Jung 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

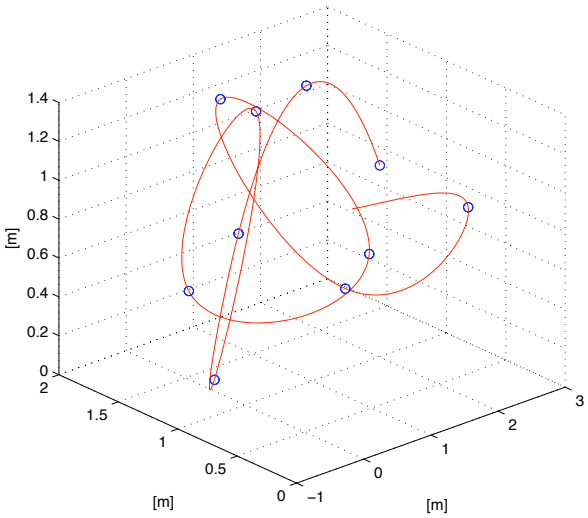


Fig. 4: The simulated 3D camera path in Matlab.

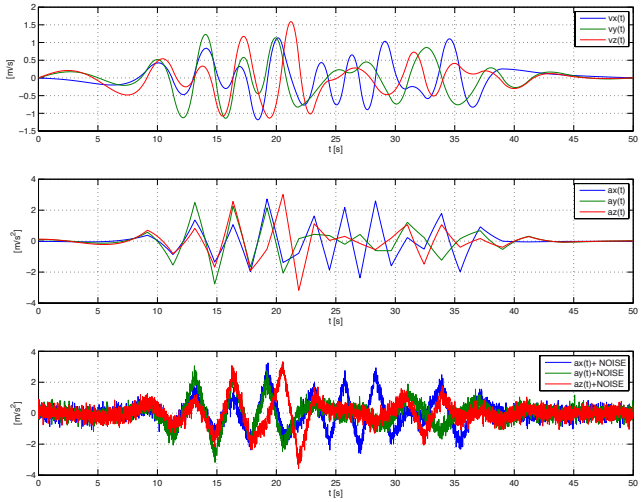


Fig. 5: The simulated velocity, acceleration and the acceleration with added noise from the 3D spline in Matlab.

(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 $\mathbf{A}$, which should be preconditioned beforehand and also to the fact that almost only the spline variables c_i are in the matrix $\mathbf{A}$ which results in bad estimation of all the other parameters. The online spline fitting approach could not be realised satisfyingly, because the performance of a normal spline fitting with 3-5 splines in a section T_s was not accurate at all. It turned out that the spline fitting task becomes more and more accurate the longer our timeline is. With up to 100 splines in 10 seconds we achieved scale estimation errors of around 5% and less. On the other hand, a duration of $T_s = 10$ seconds introduces a far too long lag into the system, what makes an online application for a controller rather critical.

V. Extended Kalman Filter

Online Extended Kalman Filter Video: URL

A. Overview

Unlike its linear counterpart, the extended Kalman filter in general is not an optimal estimator. In addition, if the initial estimate of the state is wrong, or if the process is modeled incorrectly, the filter may quickly diverge, owing to its linearisation. Another problem with the extended Kalman filter is that the estimated covariance matrix tends to underestimate the true covariance matrix and therefore risks becoming inconsistent in the statistical sense. However the extended Kalman filter can give reasonable performance mostly in conjunction with a long itterative tuning process.

B. EKF Equations

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

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

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

$$\tag{8}$$

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. Eq. 7 is a simple discrete integration over the time T, which varies. Every vector in $\vec{z}_k$ is resolved in the world frame $\mathbf{W}$. The model in its linearised form yields,

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{I}_3 & \frac{T}{\lambda}\mathbf{I}_3 & \frac{T^2}{2\lambda}\mathbf{I}_3 & -\frac{T}{\lambda^2}\mathbf{I}_3 - \frac{T^2}{2\lambda^2}\mathbf{I}_3 \\ 0 & \mathbf{I}_3 & T\mathbf{I}_3 & 0 \\ 0 & 0 & \mathbf{I}_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Different modifications in the state space model of eq. 7, 10, 11 can lead to better performance, but unfortunately there is no rule. In our case, we first had the non-linear term in the measurement equation for the vision part, which proved to be a very bad starting point. A linearisation in the measurement equation introduces approximations in every Kalman filter equation, whereas a linearisation in the state space approximates the prediction of the Kalman filter only. We also tried to substitute the non-linear term $1/\lambda = \alpha$ which yields a much easier linearised matrix $\mathbf{F}_k$ with no singularities at $\lambda = 0$.

Unfortunately, there was absolutely no performance gain. On the contrary, with α as the new parameter to be estimated, the converging was delayed for several seconds.

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

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

$$\vec{y}_{I,k} = \mathbf{H}_{I,k}\vec{z}_k = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & 0 \end{bmatrix} \vec{z}_k \quad (11)$$

The innovation for the vision part is,

$$\mathbf{K}_{V,k} = \mathbf{P}_k^- \mathbf{H}_{V,k}^\top (\mathbf{H}_{V,k}\mathbf{P}_k^- \mathbf{H}_{V,k}^\top + \mathbf{R}_V)^{-1} \quad (12)$$

$$\vec{z}_k = \vec{z}_k + \mathbf{K}_{V,k}(\vec{x}_{SLAM} - \mathbf{H}_{V,k}\vec{z}_k) \quad (13)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_{V,k}\mathbf{H}_{V,k})\mathbf{P}_k^- \quad (14)$$

The innovation for the IMU part is,

$$\mathbf{K}_{I,k} = \mathbf{P}_k^- \mathbf{H}_{I,k}^\top (\mathbf{H}_{I,k}\mathbf{P}_k^- \mathbf{H}_{I,k}^\top + \mathbf{R}_I)^{-1} \quad (15)$$

$$\vec{z}_k = \vec{z}_k + \mathbf{K}_{I,k}(\vec{a}_{IMU} - \mathbf{H}_{I,k}\vec{z}_k) \quad (16)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_{I,k}\mathbf{H}_{I,k})\mathbf{P}_k^- \quad (17)$$

The two matrices $\mathbf{R}_I, \mathbf{R}_V$ are the noise covariance matrices for the IMU and vision measurement inputs $\vec{x}_{SLAM}, \vec{a}_{IMU}$ which are resolved in the world frame $\mathbf{W}$. The vector $\vec{x}_{SLAM}$ is the position without scale obtained from the vision algorithm (SLAM). The IMU measurement $\vec{a}_{IMU}$ needs special attention, because significant errors arise in the conversion from the raw IMU output. According to Fig. 2, the acceleration in the world frame is given by,

$$_w\vec{a} = \mathbf{R}_{wc}\mathbf{R}_{ca}(_a\vec{a} - \vec{b}) - _w\vec{g} \quad (18)$$

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 initialisation of the map by the SLAM algorithm. With good initialisation the error of the angles are around ±1-2° which increases with the distance to the initialisation point (drift). This error is due to the fact that we did not adapt the camera model in the SLAM algorithm of Klein, which is still perspective and not fish-eye. 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.

C. 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 recorded the two measurement inputs for the Kalman filter, the vision non-scaled position from Georg Klein'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. Fig. 7 shows the scale estimation $\lambda(t)$ for the 3D path on the left and for the measurement set on the right for the three different setups. Always the same path and measurement set was used for the three setups. To make the measurements of the 3D path equivalent to the real data, we chose the same standard deviations for the acceleration noise of the 3D path as measured from the real data ($\sigma_{SLAM} = 0.01$ (with a $\lambda = 1 \rightarrow 0.01$ m), $\sigma_{IMU} = 0.2\ m/s^2$) The initial velocity and acceleration for the state vector were set to zero. The initial lambda 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 seconds 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, eq. 15,12 are performed as soon as the measurements arrive. The distribution over time looks as shown in Fig. 6. Before the correction we do a prediction either with T_i or T_v in the matrix $\mathbf{F}_k$.

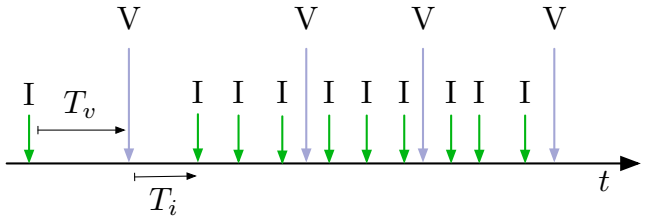
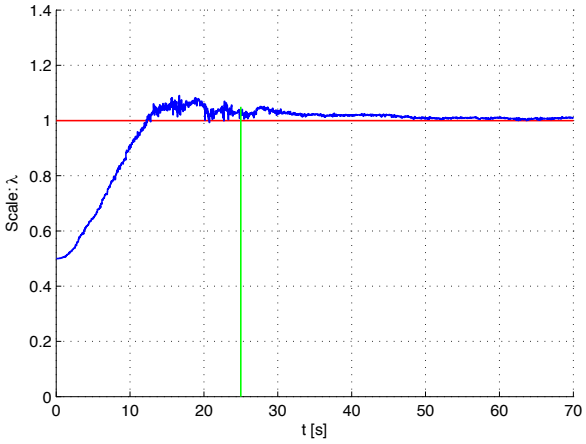


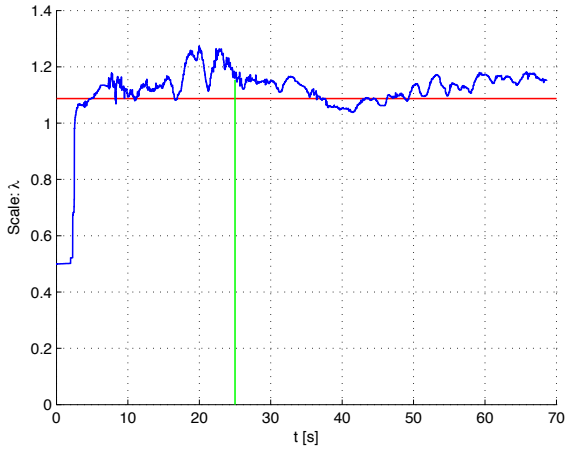
Fig. 6: Distribution of the measurements over time, (I: IMU measurement arrived, V: vision measurement arrived).

The simulation on the 3D path produces very nice results in any setup which is not suprising because we simulated the acceleration from the spline ideally and already resolved in the world frame, so eq. 18 was not needed. This is also the reason why the three plots a,c,e 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 after 70 seconds. 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. 18, 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. 7 (b). 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 (dynamic bias) on the Z acceleration.

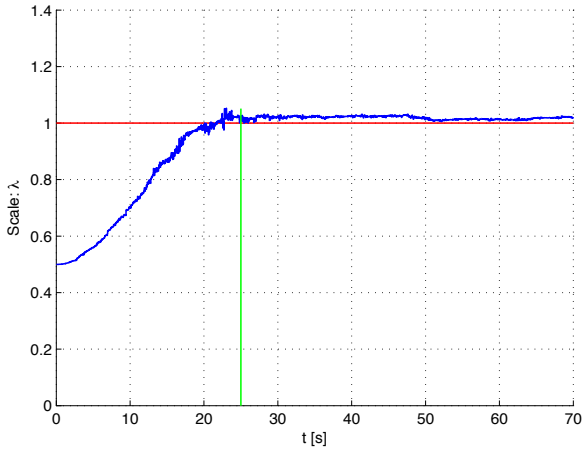
Simulation → Reality



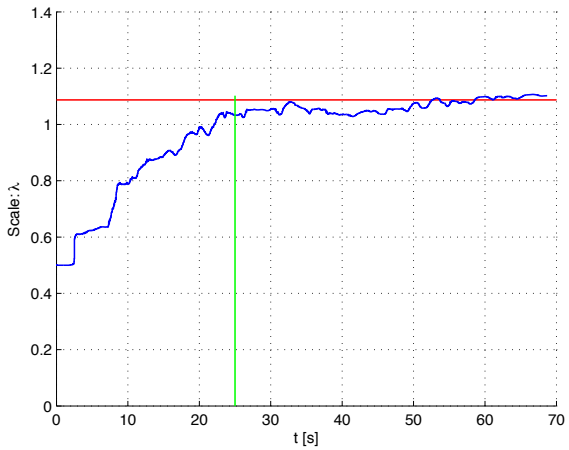
(a) X,Y,Z estimate with the 3D path



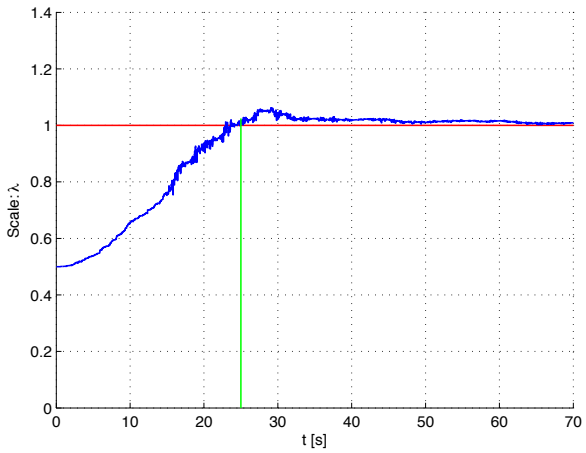
(b) X,Y,Z estimate with real data



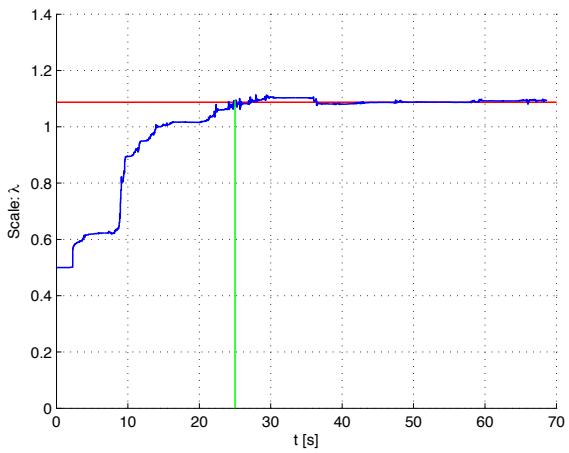
(c) X,Y estimate with the 3D path



(d) X,Y estimate with real data



(e) Z estimate with the 3D path



(f) Z estimate with real data

Fig. 7: On the left: Offline simulation of the Kalman filter with the generated 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 red line shows the fixed $\lambda = 1$ 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 after 70 seconds. 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 (dynamic bias) on the Z acceleration.

D. Online Implementation Results

Online Extended Kalman Filter Video: URL

For an online implementation, we embedded only the Z-estimation (third setup described in section C) into the code of Georg Klein [23]. The code is written in C++ and uses the computer vision library `libcvd` and the numeric library `TooN`. The workflow of our implementation is shown in Fig. 8. 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 rotation matrix $\mathbf{R}_{wc}$ of the actual camera after the initialisation of the map has been done. The MapMaker is responsible for the storage of the keyframes and executes the bundle adjustment over the whole keyframes. The Tracker also pokes the recovery algorithm when the Tracker gets lost. The recovery algorithm tries to re-find the actual camera pose by comparing the video frames to the keyframes.

We added 2 more important threads, the IMU thread and the Kalman thread itself. 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 to zero. The value for the initial λ is calculated by integration of the acceleration over 1 second beforehand.

We also introduced time-varying values into the covariance matrix $\mathbf{Q}$, 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, 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 $\mathbf{P}$ is not reset.

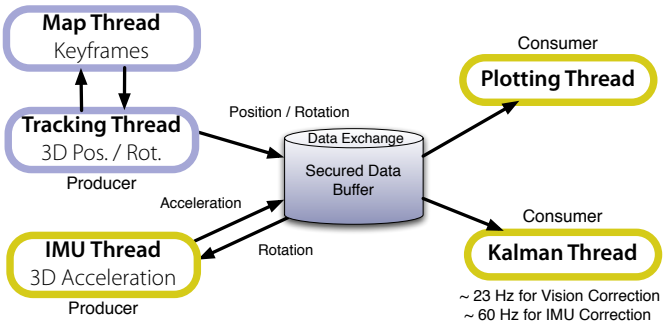


Fig. 8: Workflow of the online implementation. The original SLAM algorithm consists of two threads (blue), the Tracking thread and the MapMaker thread. We added two more threads (yellow) for the scale estimation. The IMU thread which provides the IMU acceleration data and the Kalman filter thread itself which estimates the scale factor λ.

VI. Conclusion

This paper describes 2 different approachs to estimate the absolute scale factor. The first is an online spline fitting method which proved to be not applicable for a robust online use due to the bad results of the least square optimisation. The second approach, the Kalman filter, produces good and accurate estimates within a reasonable converging time of up to 15 seconds. The online implementation works smoothly as shown in the video **??**. The Kalman filter thread which has been implemented in the SLAM algorithm of Georg Klein can now be used on an autonomous helicopter equipped with only one camera and a tri-axial accelerometer to estimate the scale factor which is very important for a trajectory reference control which will be implemented in future work.

References

[1] S.-H. Jung and C.J. Taylor. 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*, volume 2, pages II–732–II–737, 2001.

[2] Georg Klein and David Murray. 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, November 2007.

[3] Morgan Ulvklo Jonas Nygards, Per Skoglar. Navigation aided image processing in uav surveillance: Preliminary results and design of an airborne experimental system. *Journal of Robotic Systems*, 21(2):63–72, 2004.

[4] Frèdèric Labrosse. The visual compass: Performance and limitations of an appearance-based method. *Journal of Field Robotics*, 23(10):913–941, 2006.

[5] J.-C. Zufferey and D. Floreano. Fly-inspired visual steering of an ultralight indoor aircraft. *Robotics, IEEE Transactions on*, 22(1):137–146, Feb. 2006.

[6] D. Baehring, S. Simon, W. Niehsen, and C. Stiller. Detection of close cut-in and overtaking vehicles for driver assistance based on planar parallax. pages 290–295, June 2005.

[7] S. Derrouich, K. Izumida, and K. Shiiya. A combination of monocular ccd camera and inertial-sensor for range estimation. *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, 3:2191–2196 vol.3, Nov. 2002.

[8] L. Armesto, S. Chroust, M. Vincze, and J. Tornero. Multi-rate fusion with vision and inertial sensors. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 1:193–199 Vol.1, April-1 May 2004.

[9] S. G. Chroust and M. Vincze. Fusion of vision and inertial data for motion and structure estimation. *J. Robot. Syst.*, 21(2):73–83, 2004.

[10] Zaoui M. AU Wormell D., Altshuler Y., Foxlin E., and McIntyre J. A 6 d.o.f. opto-inertial tracker for virtual reality experiments in microgravity. *Acta Astronautica*, 49:451–462, August 2001.

[11] J. Eino, M. Araki, J. Takiguchi, and T. Hashizume. Development of a forward-hemispherical vision sensor for acquisition of a panoramic integration map. *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pages 76–81, Aug. 2004.

[12] D.M. Helmick, Yang Cheng, D.S. Clouse, L.H. Matthies, and S.I. Roumeliotis. Path following using visual odometry for a mars rover in high-slip environments. *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, 2:772–789 Vol.2, March 2004.

[13] D.M. Helmick, S.I. Roumeliotis, M.C. McHenry, and L. Matthies. Multi-sensor, high speed autonomous stair climbing. *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, 1:733–742 vol.1, 2002.

[14] Goldbeck J., Huertgen B., Ernst S., and Kelch L. Lane following combining vision and dgps. *Image and Vision Computing*, 18:425–433(9), 2000.

[15] Seong-Baek Kim, Seung-Yong Lee, Ji-Hoon Choi, Kyoung-Ho Choi, and Byung-Tae Jang. A bimodal approach for gps and imu integration for land vehicle applications. *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, 4:2750–2753 Vol.4, Oct. 2003.

[16] S. Niwa, T. Masuda, and Y. Sezaki. Kalman filter with time-variable gain for a multisensor fusion system. *Multisensor Fusion and Integration for Intelligent Systems, 1999. MFI '99. Proceedings. 1999 IEEE/SICE/RSJ International Conference on*, pages 56–61, 1999.

[17] Miguel Ribo, Markus Brandner, and Axel Pinz. A flexible software architecture for hybrid tracking. *J. Robot. Syst.*, 21(2):53–62, 2004.

[18] CLARK R. Robert, LIN Michael H., and TAYLOR Colin J. 3d environment capture from monocular video and inertial data. *Proceedings of SPIE, The International Society for Optical Engineering*, 2006.

[19] Irem Stratmann and Erik Solda. Omnidirectional vision and inertial clues for robot navigation. *Journal of Robotic Systems*, 21(1):33–39, 2004.

[20] J. Waldmann. Line-of-sight rate estimation and linearizing control of an imaging seeker in a tactical missile guided by proportional navigation. *Control Systems Technology, IEEE Transactions on*, 10(4):556–567, Jul 2002.

[21] A. Huster, E.W. Frew, and S.M. Rock. Relative position estimation for auvs by fusing bearing and inertial rate sensor measurements. *Oceans '02 MTS/IEEE*, 3:1863–1870 vol.3, Oct. 2002.

[22] InerVis IMU CAM Calibration Toolbox: http://www2.deec.uc.pt/~jlobo/InerVis_WebIndex/InerVis_Toolbox.html.

[23] Georg Klein. Source code of PTAM (Parallel Tracking and Mapping), http://www.robots.ox.ac.uk/~gk/PTAM/.