Fusion of Visual SLAM Pose Estimate and IMU Measurements

Michael Blösch and Timothy D. Barfoot

***Abstract*—The present paper tackles an important issue in visual SLAM algorithms: the bias on the scale of the pose estimate, which is especially important for monocular SLAM algorithms, and the possible drift of the map orientation. Based on an additionally mounted IMU, the proposed method formulates a non-linear least squares problem on a set of SLAM pose estimates together with the corresponding IMU measurements and, applying the Gauss-Newton method, estimates the scale factor and orientation of the SLAM outputs. This method can be repeatedly applied to the output of any arbitrary SLAM algorithm in order to increase its quality.**

***Index Terms*—Monocular SLAM, Observability, Map Drift, Gauss-Newton, IMU.**

I. Introduction and Related Work

IN the past years, the problem of localization and mapping within an unknown environment has been tackled by a wide range of methods. The corresponding algorithms, often called SLAM (Simultaneous Localization And Mapping) algorithms, can be based on different constellations of sensors. One very popular approach is to use a single monocular camera, as it is a sensor type producing very rich data with relatively cheap acquisition costs, low power consumption and light weight. However it brings several challenges with it. One of them is caused by the projective nature of a single camera: for arbitrary (and unknown) camera movements, the scale of the environment will never be observable.

An even more problematic issue is that, for algorithms that do not use any motion model and that do not assume any variance of the camera's acceleration, this scale can drift thoroughly within the estimated map. That means that even if the scale is correctly initialized at one point in the map, it can get a different value at an other point. This can be illustrated by an orthogonal distorted 3D grid representing the coordinate frame on which the map and the position of the camera are measured (see Fig. 1). Similarly to this scale drift, the local orientation of the map with respect to an inertial frame can also drift for all visual SLAM algorithms.

Generally, the amount of the drift remains locally small. However it can grow quite large for bigger trajectories. This can be fatal for applications that are based on the pose estimate of the SLAM algorithm. E.g., if the pose estimate is used to control an aerial vehicle it will crash after a few meters, in dependence of the intrinsic stability of the vehicle.

An approach to this problem is to include the information from an additional IMU which is rigidly fixed to the camera. In an inverse depth EKF-based SLAM algorithm Pinies et al. [1] included the IMU measurements into the motion model of the extended Kalman filter. With this they managed to limit the scale drift. However, because they do not account for a drift of the map rotation and thus the gravitational acceleration of the accelerometer might be compensated in a wrong manner, this algorithm might even yield bigger scale drift if the rotational drift between map and inertial frame gets to large.

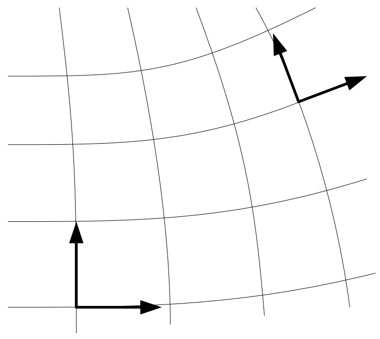
Fig. 1. Othogonal Distorted Grid. The scale, roll, and pitch drift can be illustrated by means of an orthogonal distorted grid. Locally the grid is approximately undistorted and represents the coordinate frame in which the SLAM data is evaluated. However, due to accumulated errors, for larger distances the grid can distort which leads to changing scale factors and rotations.

Another IMU based method was implemented by Lupton et al. [2]. They include the IMU observations in a delayed state information smoothing SLAM technique where all observations and states are retained separately and thereby achieved to get rid of the scale bias as well as the drift of pitch and roll angle.

In this paper, we present a method that is in some way similar to the one presented in [2]. We formulate a non-linear least squares problem on a set of SLAM pose estimates together with the corresponding IMU measurements and then apply the Gauss-Newton method in order to estimate the scale, the roll, and the pitch of the SLAM coordinate frame. In contrast to other methods we do not include the IMU observations directly into the SLAM algorithm. However, our method can be applied a posteriori to any arbitrary SLAM algorithm in order to correct the drifts. Also, in an online application the method could be executed every few meters, as the drift remains generally small for short distances.

The outline of the paper is as follows. Section II discusses the modeling of the problem and presents the method used to solve it. Section III shows the results and discusses them. And, finally, section IV states the conclusion.

II. Methodology

Given a bunch of pose estimates (orientation and position of the camera) from a SLAM algorithm and the corresponding IMU measurements, our goal is to estimate the scale and the roll and pitch angles of these estimates with respect to an inertial frame in which the z-axis is aligned with the gravitational acceleration. For this we formulate a non-linear least squares problem and use the Gauss-Newton method to solve it.

A. Notation

For a better overview, we begin by presenting the employed notation and quantities.

Common notations:

$\boldsymbol{v}_A$	Vector $\boldsymbol{v}$ expressed in the A coordinate system.
$\boldsymbol{R}_{BA}$	Rotation matrix from coordinate system A to coordinate system B.
$\boldsymbol{v}^{\times}$	Skew symmetric matrix of the vector $\boldsymbol{v}$.

Coordinate systems:

I	Inertial coordinate system, is chosen so that the gravitational acceleration is aligned with the z-axis.
S	Coordinate system in which the SLAM pose estimates are expressed.
V	Coordinate system of the vehicle. The IMU measurements are obtained in this frame.
C	Coordinate system of the camera.

Vectors and scalars:

$\boldsymbol{r}_k := \boldsymbol{r}_i^{v_k i}$	Position vector of the vehicle expressed in the inertial frame.
$\boldsymbol{a}_k := \ddot{\boldsymbol{r}}_i^{v_k i}$	Acceleration of the vehicle expressed in the inertial frame.
$\boldsymbol{\omega}_k := \boldsymbol{\omega}_i^{v_k i}$	Angular speed of the vehicle frame with respect to the inertial frame.
$\boldsymbol{f}_k := \boldsymbol{f}_{v_k}^{v_k i}$	Specific force measured by the accelerometer expressed in the vehicle frame.
$\boldsymbol{s}_k := \lambda \boldsymbol{s}_s^{c_k s}$	Position estimate of the SLAM algorithm expressed in the SLAM frame.
λ	Scale factor between the SLAM position estimates and the true positions.
T_k	Elapsed time between step k and $k+1$

Matrices:

$\boldsymbol{C}_k := \boldsymbol{R}_{v_k i}$	Rotation from inertial frame to vehicle frame.
$\boldsymbol{O} := \boldsymbol{R}_{si}$	Rotation from inertial frame to SLAM frame.

Constant parameters:

$\boldsymbol{g} := \boldsymbol{g}_i$	Gravitational acceleration expressed in the inertial frame.
$\boldsymbol{\rho} := \boldsymbol{\rho}_{v_k}^{c_k v_k}$	Distance between vehicle and camera expressed in the vehicle frame.
$\boldsymbol{V} := \boldsymbol{R}_{c_k v_k}$	Rotation from vehicle frame to camera frame.

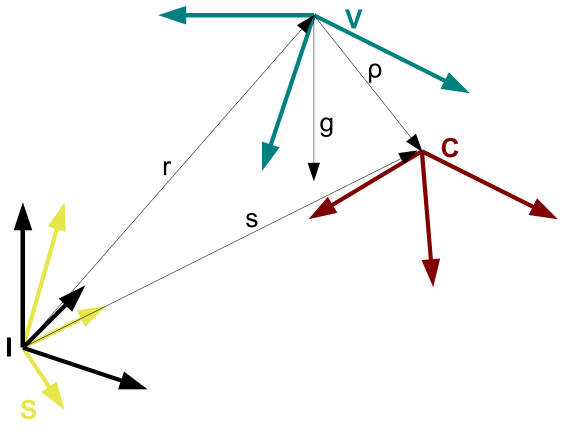

Fig. 2. In this figure the four coordinate frames can be observed. I is the inertial coordinate frame. V is the vehicle fixed coordinate frame, in which also the IMU measurements are expressed. C is the camera fixed coordinate frame. S is the coordinate frame in which the SLAM measurements are expressed.

In Fig. 2 the different coordinate frames and quantities are displayed. In this approach, rotations are represented by their rotation matrices. However, noise or iterative corrections will be represented by rotation vectors and will have the form of small disturbances:

$$\boldsymbol{R} = (\boldsymbol{1} - \boldsymbol{d}^{\times})\bar{\boldsymbol{R}}$$

B. Motion Model and Measurement Model

The motion model uses the IMU measurements in order to describe the relation between subsequent pose estimate. For this a discrete derivative of the pose estimates is computed:

$$\boldsymbol{a}_k = \frac{\boldsymbol{r}_{k+1} - (1 + \frac{T_k}{T_{k-1}})\boldsymbol{r}_k + \frac{T_k}{T_{k-1}}\boldsymbol{r}_{k-1}}{\frac{1}{2}T_k(T_k + T_{k+1})} \quad (1)$$

$$\boldsymbol{\Psi}_k = \boldsymbol{C}_{k+1}\boldsymbol{C}_k^T \quad (2)$$

where the following intermediate results are used

$$\boldsymbol{a}_k = \boldsymbol{g} - \boldsymbol{C}_k^T(\boldsymbol{f}_k + \boldsymbol{w}_k^f) \quad (3)$$

$$\boldsymbol{\psi}_k := T_k(\boldsymbol{\omega}_k + \boldsymbol{w}_k^{\omega}) \quad (4)$$

$$\boldsymbol{\Psi}_k := cos\,|\boldsymbol{\psi}_k|\,\boldsymbol{1} + (1 - cos\,|\boldsymbol{\psi}_k|)\left(\frac{\boldsymbol{\psi}_k}{|\boldsymbol{\psi}_k|}\right)\left(\frac{\boldsymbol{\psi}_k}{|\boldsymbol{\psi}_k|}\right)^T - sin\,|\boldsymbol{\psi}_k|\left(\frac{\boldsymbol{\psi}_k}{|\boldsymbol{\psi}_k|}\right)^{\times} \quad (5)$$

and where $\boldsymbol{w}_k^f$ and $\boldsymbol{w}_k^{\omega}$ represent the noise on the IMU measurements.

As the estimation method will be applied on relatively short time span, we assume that the scale, roll, and pitch biases are constant over the set of SLAM pose estimates. Thus we can formulate the measurement model as follows:

$$\boldsymbol{s}_k = \lambda\boldsymbol{O}\left(\boldsymbol{r}_k + \boldsymbol{C}_k^T\boldsymbol{\rho}\right) + \boldsymbol{n}_k^s \quad (6)$$

$$\boldsymbol{R}_k = (\boldsymbol{1} - \boldsymbol{n}_k^{R\times})^T\boldsymbol{V}\boldsymbol{C}_k\boldsymbol{O}^T \quad (7)$$

Here, $\boldsymbol{n}_k^s$ and $\boldsymbol{n}_k^R$ represent the noise on the SLAM pose estimates.

C. Unobservability of the Yaw Bias

One interesting point, that can be derived from the motion and measurement models, is that the equations are invariant when applying a rotation around the z-axis. Let $\boldsymbol{R}_z$ be an arbitrary rotation around the z-axis and $\boldsymbol{r}_k$, $\boldsymbol{C}_k$, λ, $\boldsymbol{O}$ be a solution of the set of equations (1)-(7), then

$$\boldsymbol{r}'_k = \boldsymbol{R}_z^T \boldsymbol{r}_k, \quad \boldsymbol{C}'_k = \boldsymbol{C}_k \boldsymbol{R}_z, \quad \boldsymbol{O}' = \boldsymbol{O}\boldsymbol{R}_z, \quad \lambda' = \lambda$$

will also satisfy the set of equations. The reason for this is, that the gravitational acceleration is invariant under rotation around the z-axis:

$$\boldsymbol{g} = \boldsymbol{R}_z^T \boldsymbol{g}$$

D. Gauss-Newton Implementation

The Gauss-Newton method linearizes the errors around the actual estimate and iteratively updates the estimate. In order to linearize rotational errors we use the approximation by small disturbances. We introduce the following disturbances:

$$\begin{aligned} &\boldsymbol{r}_k = \bar{\boldsymbol{r}}_k + \delta\boldsymbol{r}_k \\ &\boldsymbol{C}_k = (\boldsymbol{1} - \delta\boldsymbol{\phi}_k{}^{\times})\bar{\boldsymbol{C}}_k \\ &\lambda = \bar{\lambda} + \delta\lambda \\ &\boldsymbol{O} = (\boldsymbol{1} - \delta\boldsymbol{o}^{\times})\bar{\boldsymbol{O}} \end{aligned} \tag{8}$$

With this the linearization at one time step k yields

$$\boldsymbol{e}_k = \bar{\boldsymbol{e}}_k + \boldsymbol{H}_k \begin{bmatrix} \delta\boldsymbol{x}_{k-1} \\ \delta\boldsymbol{x}_k \\ \delta\boldsymbol{x}_{k+1} \end{bmatrix} + \boldsymbol{G}_k \begin{bmatrix} \delta\lambda \\ \delta\boldsymbol{o} \end{bmatrix} \tag{9}$$

where $\delta\boldsymbol{x}_k = [\delta\boldsymbol{r}_k^T, \delta\boldsymbol{\phi}_k^T]^T$ is the disturbance on the kth state. $\boldsymbol{H}_k$ and $\boldsymbol{G}_k$ are the Jacobian matrices that can be computed as follows:

$$\boldsymbol{H}_k = \begin{bmatrix} -\frac{T_k}{T_{k-1}} & 0 & \frac{T_k+T_{k-1}}{T_{k-1}} & T_k\frac{T_k+T_{k-1}}{2}\boldsymbol{C}_k^T\boldsymbol{f}_k^{\times} & -\boldsymbol{1} & 0 \\ 0 & -\boldsymbol{\Psi}_{k-1} & 0 & \boldsymbol{1} & 0 & 0 \\ 0 & 0 & -\lambda\boldsymbol{O} & \lambda\boldsymbol{O}\boldsymbol{C}_k^T\boldsymbol{\rho}^{\times} & 0 & 0 \\ 0 & 0 & 0 & -\boldsymbol{V} & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{G}_k = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\boldsymbol{O}\boldsymbol{r}_k & -\lambda(\boldsymbol{O}\boldsymbol{r}_k)^{\times} \\ 0 & \boldsymbol{R}_k \end{bmatrix}$$

The mean error $\bar{\boldsymbol{e}}_k$ can be directly obtained from the model

$$\bar{\boldsymbol{e}}_k = \begin{bmatrix} T_k\frac{T_k+T_{k-1}}{2}\boldsymbol{a}_k - \boldsymbol{r}_{k+1} + \frac{T_k+T_{k-1}}{T_{k-1}}\boldsymbol{r}_k - \frac{T_k}{T_{k-1}}\boldsymbol{r}_{k-1} \\ \boldsymbol{1} - \boldsymbol{C}_k\boldsymbol{C}_{k-1}^T\boldsymbol{\Psi}_{k-1}^T \\ \boldsymbol{s}_k - \lambda\boldsymbol{O}\left(\boldsymbol{r}_k + \boldsymbol{C}_k^T\boldsymbol{\rho}\right) \\ \boldsymbol{1} - \boldsymbol{R}_k\boldsymbol{O}\boldsymbol{C}_k^T\boldsymbol{V}^T \end{bmatrix}$$

and the corresponding covariance matrix for a correct error weighting is given by

$$\boldsymbol{T}_k = diag\begin{pmatrix} \left(T_k\frac{T_k+T_{k-1}}{2}\right)^2 \boldsymbol{C}_k^T Cov\left[\boldsymbol{w}_k^f\right]\boldsymbol{C}_k \\ T_{k-1}^2 Cov\left[\boldsymbol{w}_k^{\omega}\right] \\ Cov\left[\boldsymbol{w}_k^s\right] \\ Cov\left[\boldsymbol{w}_k^R\right] \end{pmatrix}$$

Please note that this is the error for a general case. In some cases IMU measurements or SLAM measurements might not be available and the equations have to be adapted adequately. By subsequently stacking the errors together, we get an equation for the total error.

$$\boldsymbol{e} = \bar{\boldsymbol{e}} + \boldsymbol{H}\delta\boldsymbol{x} \tag{10}$$

with $\delta\boldsymbol{x} = [\delta\boldsymbol{x}_1^T, \ldots, \delta\boldsymbol{x}_K^T, \delta\lambda, \delta\boldsymbol{o}^T]^T$, $\bar{\boldsymbol{e}} = [\delta\bar{\boldsymbol{e}}_1^T, \ldots, \delta\bar{\boldsymbol{e}}_K^T]^T$ and where $\boldsymbol{H}$ has to be correctly constructed from the matrices $\boldsymbol{H}_1$ to $\boldsymbol{H}_K$ and $\boldsymbol{G}_1$ to $\boldsymbol{G}_K$. The corresponding covariance matrix is then given by

$$\boldsymbol{T} = diag(\boldsymbol{T}_1, \ldots, \boldsymbol{T}_K)$$

So, given an estimate, a correction step can be performed by solving the following equation for the state disturbance $\delta\boldsymbol{x}$

$$(\boldsymbol{H}^T\boldsymbol{T}^{-1}\boldsymbol{H})\delta\boldsymbol{x} = -\boldsymbol{H}^T\boldsymbol{T}^{-1}\bar{\boldsymbol{e}} \tag{11}$$

and then computing a better estimate by using the equations (8).

E. Initial Guess

In order to produce an initial guess, an estimate for the rotation $\boldsymbol{O}$ is computed based on the fact that for small accelerations $\boldsymbol{a}_k$ we have

$$\boldsymbol{C}_k^T\boldsymbol{f}_k \approx \boldsymbol{g} \tag{12}$$

This can be transformed into

$$\boldsymbol{O}^T\boldsymbol{R}_k^T\boldsymbol{V}\boldsymbol{f}_k \approx \boldsymbol{g} \tag{13}$$

which can not be satisfied in general. However, it is possible to find a rotation $\boldsymbol{O}$ so that the vectors on both side of the equation point into the same direction (what is equivalent to the least square solution). This gives us two constraint on the rotation $\boldsymbol{O}$, with which we are able to determine the roll and the pitch of $\boldsymbol{O}$.

With the initial guess for $\boldsymbol{O}$ and an arbitrary initial guess for λ, the initial guesses for $\boldsymbol{r}_k$ and $\boldsymbol{C}_k$ can be produced based on the SLAM measurements (where we may have to interpolate some data if the measurements are asynchronous):

$$\boldsymbol{C}_k = \boldsymbol{V}^T\boldsymbol{R}_k\boldsymbol{O} \tag{14}$$

$$\boldsymbol{r}_k = \frac{1}{\lambda}\boldsymbol{O}^T\boldsymbol{s}_k - \boldsymbol{C}_k^T\boldsymbol{\rho} \tag{15}$$

III. Results and Discussion

The data is simulated based on a real trajectory. However, only the rotational speed measurements are obtained from a real IMU. The specific force measurements are simulated using the discrete derivative (1) and the SLAM measurement are simulated based on equations (6) and (7). In order to simulate the noise, Gaussian noise was assumed. The values for the employed standard deviation were 0.3 for $\boldsymbol{w}_k^f$, 0.2 for $\boldsymbol{n}_k^s$ and 0.05 for $\boldsymbol{n}_k^R$.

In order to evaluate the quality of the estimate, the obtained result were subsequently corrected by the unobservable yaw drift. A typical result for a 20 seconds long trajectory with enough acceleration is displayed in the figures Fig. 3-5. Fig.

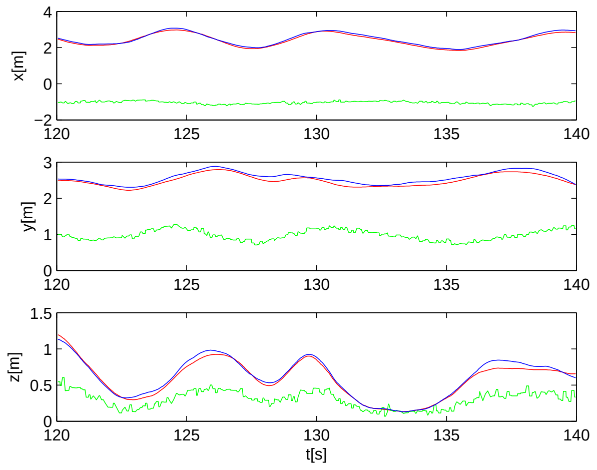

Fig. 3. Position Trajectory. Red: true, blue: estimated, green: initial guess. The estimated trajectory fits very well to the true one. The main deviation between both is caused by the scale error. The true scale is 2, while the estimated scale is 1.9518.

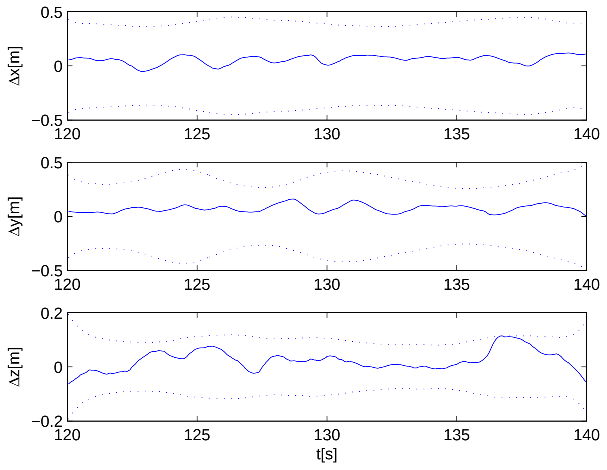

Fig. 4. Position error between true trajectory and estimated trajectory. The error nicely remains within the 3σ standard deviation envelope. The reason why the envelope seems under confident is the incertitude on the scale factor, which increases the incertitude of the position estimate.

3 shows the trajectory of the vehicle. The estimated trajectory fits very well the true trajectory, although the initial guess for the true $\lambda = 2$ was 5. The estimated scale factor converged to 1.9518 after 12 iterations in 4.3095 seconds. The estimates of the roll and pitch angle are extremely precise with errors of 0.0004 rad and -0.0012 rad. The predicted standard deviances for the roll and pitch angle (0.004 rad) as well as for the scale factor (0.1229) are very consistent with the obtained result.

In Fig. 4 and Fig. 5 the position error and the rotational error together with their 3σ covariance envelope are displayed. Also here we can argue that the estimate is consistent, the errors do only very rarely leave their envelope.

In Fig. 6 a histogram of the relative scale error over 40 different simulated datasets, based on the trajectory displayed in Fig. 3, can be observed. Depending on the simulated noise the estimate scale varies. However, its mean is near to 0 and the standard deviation remains relatively small (0.0624).

The quality of the estimate depends strongly on the underlying trajectory. Longer datasets decrease the variance of the obtained scale estimation. Fig. 7 represents the relation between the length of the dataset in seconds and the obtained standard deviation of the relative scale error. From this we can argue that a dataset length of around 20 seconds is optimal. Longer datasets do not yield any additional precision and only cost more computation, which increase quadratically.

Another factor playing an important role is the amount of acceleration in the dataset. In the worst case, when there is no acceleration, the scale can not be estimated at all. In Fig. 8 the standard deviation of the relative scale error is plotted versus the amount of mean absolute acceleration. When more acceleration is present the accuracy of the scale estimate increases. So, in order to get a satisfyingly accurate estimate the trajectory should exhibit enough acceleration (3 times the noise on the accelerometer is a good factor).

A remaining issue of this algorithm is that, if a part of the trajectory is constant (zero acceleration), a bias on the scale

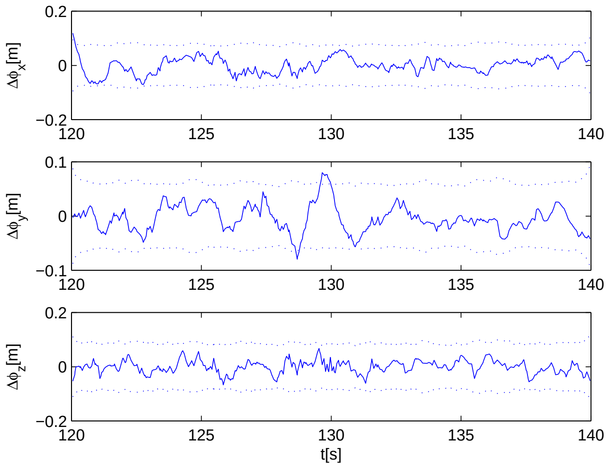

Fig. 5. Error of the estimated vehicle orientation. Also here, the error nicely remains most of the time within the 3σ standard deviation envelope.

estimate will remain. This is due to the fact that the algorithm tries to fit the noise of the accelerometer and will therefor return smaller scale estimate. Several attempt to estimate this bias based on some trajectory parameters were unsuccessful. In particular the attempt to find a relation between relative scale bias and the proportion of constant trajectory and the mean absolute acceleration failed (they were estimated by building a mixture model of chi-square distributions, which was quite accurate). In Fig. 9 the mean relative scale error is plotted versus the proportion of non-constant trajectory. In this plot a relation could be derived, however the trajectories all had similar mean absolute acceleration values. If this is not the case the relation gets much more complex.

For this application, the Gauss-Newton method convergences quality is very good. The estimate is independent of the initial guess and converges even if the initial guess of the scale factor is several factors away (can be a factor 10 aside

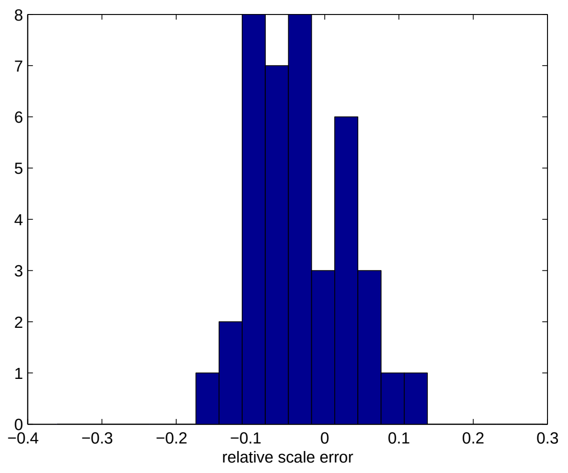

Fig. 6. Histogram built by 40 simulated datasets based on the same trajectory of 20 seconds length. The mean of the relative scale error is -0.0332 and its standard deviation is 0.0624.

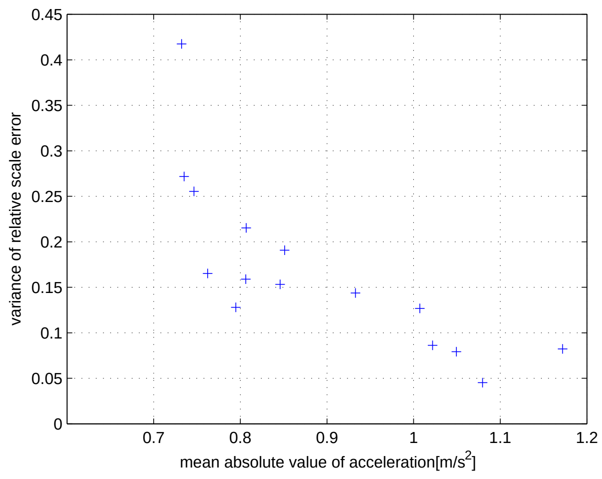

Fig. 8. Relation between the mean absolute acceleration of the data and the standard deviation of the relative scale error. A sufficiently large amount of acceleration is necessary in order to obtain an accurate scale estimate. A factor 3 between standard deviation of the noise and the mean absolute acceleration is good.

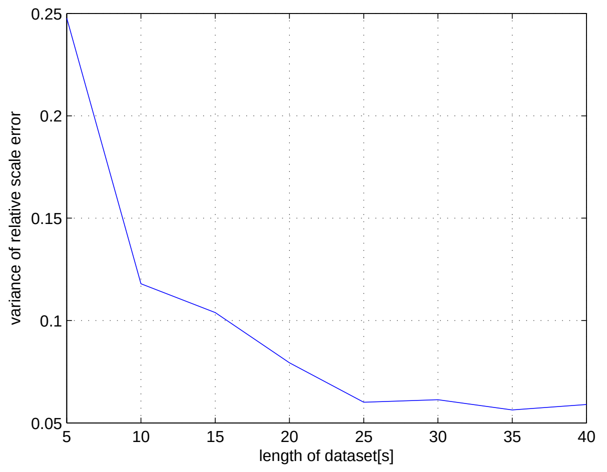

Fig. 7. Relation between length of the dataset in seconds and the obtained standard deviation of the relative scale error. Thus, longer datasets increase the accuracy of the estimated scale.

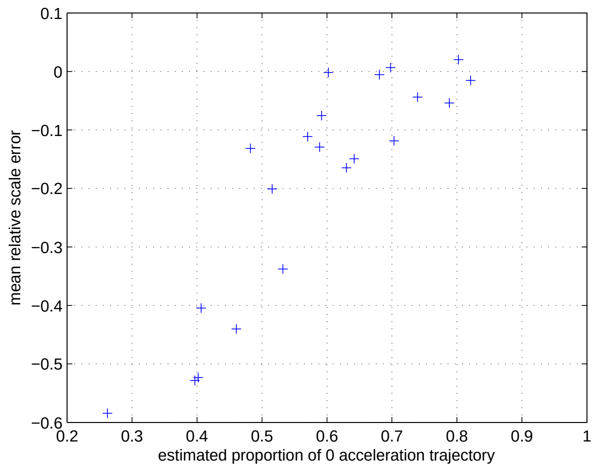

Fig. 9. The x-axis represents the proportion of the trajectory that has non-zero acceleration and the y-axis represents the mean relative scale error (scale bias). In this figure the relation between both looks quite simple. However, most points in this figure were created with similar mean absolute acceleration values.

of the true scale and still converge). Also, for a dataset of 20 seconds a good estimate can be obtained in 2-3 seconds. Thus the method could be repeatedly applied to a running SLAM algorithm in order to update the scale factor and the roll and pitch angles.

However, a remaining correction to do before this algorithm could actually be applied to a real problem, would be to include the IMU biases into the estimation process. With this extension the algorithm would probably need longer datasets in order to get the same accuracy of the scale estimate, but it would be applicable to real data. A last point to note is that the drift around the z-axis of the inertial frame is unobservable. Fortunately for most applications this drift is not of great importance.

IV. Conclusion

Based on an additional IMU, the presented method estimates the scale bias and the roll and pitch rotation drift of an arbitrary SLAM algorithm. By formulating a non-linear least square problem over a bunch of SLAM poses, the method successfully obtains an accurate estimate in the vast majority of the cases. Problems mainly occur when there is not enough motion in the trajectory, which is a more intrinsic issue.

Results on simulated data support the method's performance. It could be applied a posteriori to any arbitrary SLAM algorithm that produces position and rotation estimates. However for this, one extension would be necessary: the bias of the IMU must be taken into the estimation process. Also, for most SLAM algorithm a direct integration of the IMU measurements into the SLAM framework would be the better solution. But, for algorithms where this might not be possible or is combined with additional issues, the presented method could be used parallel to the SLAM algorithm in order to keep the scale, roll and pitch drift limited.

References

[1] P. Pinies, T. Lupton, S. Sukkarieh, and J. Tardos, "Inertial aiding of inverse depth slam using a monocular camera," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 2797–2802.

[2] T. Lupton and S. Sukkarieh, "Efficient integration of inertial observations into visual slam without initialization," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct. 2009, pp. 1547–1552.