Scale drift-aware large scale monocular SLAM

Conference Paper · June 2010

DOI: 10.15607/RSS.2010.VI.010 · Source: DBLP

J. M. M. Montiel
University of Zaragoza
67 PUBLICATIONS **3,241** CITATIONS

Scale Drift-Aware Large Scale Monocular SLAM

Hauke Strasdat
Department of Computing,
Imperial College London, UK
Email: strasdat@doc.ic.ac.uk

J.M.M. Montiel
Instituto de Investigación
en Ingeniería de Aragón (I3A),
Universidad de Zaragoza, Spain
Email: josemari@unizar.es

Andrew J. Davison
Department of Computing,
Imperial College London, UK
Email: ajd@doc.ic.ac.uk

***Abstract*— State of the art visual SLAM systems have recently been presented which are capable of accurate, large-scale and real-time performance, but most of these require stereo vision. Important application areas in robotics and beyond open up if similar performance can be demonstrated using monocular vision, since a single camera will always be cheaper, more compact and easier to calibrate than a multi-camera rig.**

With high quality estimation, a single camera moving through a static scene of course effectively provides its own stereo geometry via frames distributed over time. However, a classic issue with monocular visual SLAM is that due to the purely projective nature of a single camera, motion estimates and map structure can only be recovered up to scale. Without the known inter-camera distance of a stereo rig to serve as an anchor, the scale of locally constructed map portions and the corresponding motion estimates is therefore liable to drift over time.

In this paper we describe a new near real-time visual SLAM system which adopts the continuous keyframe optimisation approach of the best current stereo systems, but accounts for the additional challenges presented by monocular input. In particular, we present a new pose-graph optimisation technique which allows for the efficient correction of rotation, translation and scale drift at loop closures. Especially, we describe the Lie group of similarity transformations and its relation to the corresponding Lie algebra. We also present in detail the system's new image processing front-end which is able accurately to track hundreds of features per frame, and a filter-based approach for feature initialisation within keyframe-based SLAM. Our approach is proven via large-scale simulation and real-world experiments where a camera completes large looped trajectories.

I. Introduction

Visual SLAM with a single camera is more challenging than when stereo vision can be used, but successful solutions have the potential to make a much wider impact because of the wealth of application domains in robotics and beyond where a single camera can more cheaply, compactly and conveniently be installed. Monocular SLAM research, which naturally overlaps with the highly developed Structure from Motion (SfM) field in computer vision, has seen great progress over the last ten years, and several impressive real-time systems have emerged as researchers have managed to transfer the best sequential SLAM techniques to the single camera domain. However, the successful sequential systems presented have mostly either been limited to operation in a local workspace (e.g. [7, 15, 9, 2]) or, more rarely, have been designed for open-loop visual odometry exploration of larger areas [20, 3].

Meanwhile, in stereo vision-based SLAM there have now been systems presented which offer end to end solutions to real-time large scale SLAM. Probably the most complete is that of Mei *et al.* [17], which combines robust and accurate local visual odometry with constant-time large-scale mapping (enabled by a continuous relative map representation), high-performance appearance-based loop closure detection, and global metric map optimisation if required. Another similar system is that presented by Konolige and Agrawal [16].

It has proven more difficult to achieve real-time large-scale mapping with a monocular camera, due to its nature as a purely projective sensor. Geometry does not just 'pop out' of the data from a moving camera, but must be inferred over time from multiple images. Difficulties had to be overcome before a probabilistic sequential approach could successfully be implemented for monocular SLAM due to the fact that image features must be observed from multiple viewpoints with parallax before fully constrained 3D landmark estimates can be made. Special attention was given to feature initialisation schemes which permitted sequential probabilistic filtering of the joint camera and map state [7, 27, 19]. Beyond this issue, the fact that a single camera does not measure metric scale means that monocular mapping must either proceed with scale as an undetermined factor or that scale has to be introduced by an additional information source such as a calibration object as in [7] or even by exploiting nonholonomic motion constraints [26]. These issues, and the fact that monocular maps are just generally less well constrained than those built by metric sensors, meant that it was more difficult than in SLAM systems with other sensors to build algorithms which could tackle large areas by composing local fragments.

One of the first attempts to put a whole large scale monocular SLAM together was by Clemente *et al.* [4], who took a hierarchical mapping approach where locally filtered submaps were built and joined by an upper level joint map. Map matching based on feature correspondences between submaps allowed global correction of motion and scale drift around a large loop. In [23] a more accurate later evolution of this work based on conditional independent local maps was presented. Eade and Drummond [10, 8] presented a system which had several similarities to these approaches, but also many improvements, in that it was able to operate automatically in true real-time, incorporating effective loop-closing and re-localisation methods, as well as making use of non-linear optimisation techniques at various different levels to improve accuracy. At its core, though, it built a graph of local filtered submaps, each with a state vector and covariance, and with the similarity transforms between neighbouring submaps estimated via feature correspondences.

A. SLAM via Optimisation

Recently, Strasdat *et al.* [28] presented results which indicate, on the basis of a systematic investigation into the accuracy of local map components versus computational cost, that the keyframes + optimisation approach of Klein and Murray [15] — based ultimately on well known bundle adjustment methods [29] — is strongly advantageous compared to methods like those of Eade [8] or Clemente [4] whose building blocks are locally filtered submaps. Essentially, the accuracy analysis in [28] shows that it is the ability to harvest measurements from large numbers of image features per frame that plays the major role in increasing accuracy in local monocular motion estimation, while incorporating information from a large number of closely spaced camera frames provides less information for a given computation budget. This plays into the hands of the optimisation methods which cope much more efficiently with large numbers of features than filters can.

In light of this analysis, the previous large scale monocular systems [4] and [8] can be seen as somewhat unsatisfactory approaches, which combine filtering at a local level with optimisation at the global level. For instance, when loop closures are imposed, the relative positions of their filtered submaps changes, but any drift within the local maps themselves cannot be resolved. In this paper, we have therefore resolved to take a keyframe optimisation approach from top to bottom, aiming for maximum accuracy while taking into full account the special character of SLAM with monocular vision.

B. Gauge Freedoms, Monocular SLAM and Scale Drift

Metric SLAM systems aim to build coherent maps, in a single coordinate frame, of the areas that a robot moves through. But they must normally do this based on purely relative measurements of the locations of scene entities observable by their on-board sensors. There will therefore always be certain degrees of *gauge freedom* in the maps that they create, even when the best possible job is done of estimation. These gauge freedoms are degrees of transformation freedom through which the whole map, consisting of feature and robot position estimates taken together, can be transformed without affecting the values of the sensor measurements. In SLAM performed by a robot moving on a ground plane and equipped with a range-bearing sensor, there are three degrees of gauge freedom, since the location of the entire map with regard to translations and rotation in the plane is undetermined by the sensor measurements. In SLAM by a robot moving in 3D and equipped with a sensor like calibrated stereo vision or a 3D laser range-finder, there are six degrees of gauge freedom, since the whole map could experience a rigid body transformation in 3D space. In monocular SLAM, however, there are fundamentally seven degrees of gauge freedom [29], since the overall scale of the map, as well as a 6 DoF (degrees of freedom) rigid transformation, is undetermined (scale and a rigid transformation taken together are often known as a similarity transformation).

It is the number of gauge degrees of freedom in a particular type of SLAM which therefore determines the ways in which drift will inevitably occur between different fragments of a map. Consider two distant fragments in a large map built continuously by a single robot: local measurements in each of the fragments have no effect in pulling either towards a particular location in the degrees of gauge freedom. If they are not too distant from each other, they will share some coherence in these degrees of freedom, but only via compounded local measurements along the chain of fragments connecting them. The amount of drift in each of these degrees of freedom will grow depending on the distance between the fragments, and the distribution of the potential drift can be calculated if needed by uncertainty propagation along the chain.

It is very well known that planar maps built by a ground-based robot drift in three degrees of freedom; so maps built by a monocular camera with no additional information drift in seven degrees of freedom. It is through these degrees of freedom therefore which loop closure optimisations must adjust local map fragments (poses or submaps in a graph).

II. General Framework for State Estimation using Optimisation

In this section, we give a short review of state estimation using optimisation. The goals of this are first to define common notation and terminology for the rest of the paper, and second to clarify the natural close relationship between the different types of optimisation our approach uses, such as bundle adjustment and pose-graph optimisation.

In a general estimation problem, we would like to estimate a vector of parameters $\mathbf{p}$ given a vector of measurements $\mathbf{f}$, where we know the form of the likelihood function $p(\mathbf{f}|\mathbf{p})$. The most probable solution is the set of values $\mathbf{p}$ which maximises this likelihood, which is equivalent to minimising the negative log-likelihood:

$$\arg\max_{\mathbf{p}} p(\mathbf{f}|\mathbf{p}) = \arg\min_{\mathbf{p}}(-\log p(\mathbf{f}|\mathbf{p})) \ . \tag{1}$$

In many optimisation problems, negative log-likelihood is known as *energy* and the goal is to minimise it, which is fully equivalent to maximising the probability of the solution.

A. Least Square Problems and Gauss-Newton Optimisation

We now speak in more specific but still very widely applicable terms, and assume that the likelihood distribution $p(\mathbf{f}|\mathbf{p})$ is jointly Gaussian, and thus has the distribution (up to a constant of proportionality):

$$p(\mathbf{f}|\mathbf{p}) \propto \exp((\mathbf{f} - \hat{\mathbf{f}}(\mathbf{p}))^\top \Lambda_\mathbf{f}(\mathbf{f} - \hat{\mathbf{f}}(\mathbf{p}))) \ , \tag{2}$$

where $\Lambda_\mathbf{f} = \Sigma_\mathbf{f}^{-1}$, the *information matrix* or inverse of the *measurement covariance matrix* $\Sigma_\mathbf{f}$ of the likelihood distribution, and $\hat{\mathbf{f}}(\mathbf{p})$ is the 'measurement function' which computes (predicts) the distribution of measurements $\mathbf{f}$ given a set of parameters $\mathbf{p}$.

The negative log-likelihood, or energy, $S(\mathbf{p}) = -\log p(\mathbf{f}|\mathbf{p})$ therefore has the quadratic form:

$$S(\mathbf{p}) = (\mathbf{f} - \hat{\mathbf{f}}(\mathbf{p}))^\top \Lambda_\mathbf{f} (\mathbf{f} - \hat{\mathbf{f}}(\mathbf{p})) \ . \tag{3}$$

If $\Lambda_\mathbf{f}$ is block-diagonal, $S(\mathbf{p})$ simplifies to:

$$S(\mathbf{p}) = \sum_i (\mathbf{f}_i - \hat{\mathbf{f}}_i(\mathbf{p}_i))^\top \Lambda_{\mathbf{f},i} (\mathbf{f}_i - \hat{\mathbf{f}}_i(\mathbf{p}_i)) \ . \tag{4}$$

This is the case if $\mathbf{f}$ consists of a stack of measurements which are independent, which can very commonly be assumed when many measurements are made from a calibrated sensor — of particular interest here, where each measurement is the image location of a scene feature in one frame taken by a camera.

Further, and again in our case, $\Lambda_\mathbf{f}$ can very often be taken to equal the identity matrix. This implies that the uncertainty in each individual parameter of every measurement has exactly the same value, and that they are all independent. The absolute magnitude of the values on the diagonal of $\Lambda_\mathbf{f}$ does not affect the optimisation result so they can be set to one. Now, $S(\mathbf{p})$ is simply a sum of squares, and minimising it is called *non-linear least squares optimisation*.

A common technique for non-linear least squares optimisation is the Gauss-Newton method, which is an approximation of the Newton method. This consists of iteratively updating an initial estimate of the parameter vector $\mathbf{p}$ by the rule

$$\mathbf{p}_{(i+1)} = \mathbf{p}_{(i)} + \delta \ , \tag{5}$$

where at each step the update vector δ is found by solving the *normal equation*:

$$\left(\mathtt{J}_\mathbf{p}^\top \Lambda_\mathbf{f} \mathtt{J}_\mathbf{p}\right) \delta = -\mathtt{J}_\mathbf{p}^\top \Lambda_\mathbf{f} \mathbf{r} \ . \tag{6}$$

Here, $\mathtt{J}_\mathbf{p} = \frac{\partial \mathbf{r}}{\partial \mathbf{p}}$ and $\mathbf{r} = \mathbf{f} - \hat{\mathbf{f}}(\mathbf{p})$ is the *residual error*.

All estimation described in this paper is done using a variant of Gauss-Newton called Levenberg-Marquardt (LM), which alters the normal equation as follows:

$$\left(\mathtt{J}_\mathbf{p}^\top \Lambda_\mathbf{f} \mathtt{J}_\mathbf{p} + \mu \mathtt{I}\right) \delta = -\mathtt{J}_\mathbf{p}^\top \Lambda_\mathbf{f} \mathbf{r} \ . \tag{7}$$

The parameter μ rotates the update vector δ towards the direction of the steepest descent. Thus, if $\mu \to 0$ pure Gauss-Newton is preformed, whereas if $\mu \to \infty$, *gradient descent* is used. In LM, we only perform an update $\mathbf{p}_{(i)} + \delta$ if it will be successful in significantly reducing the residual error. After a successful update, μ is decreased; otherwise μ is increased. The motivation behind this heuristic is that Gauss-Newton can approach a quadratic rate of convergence, but gradient descent behaves much better far from the minimum.

B. Covariance Back-propagation

So far, we have considered only how to find the single most probable set of parameters $\mathbf{p}$. If there is no gauge freedom in the minimisation $\arg\min_\mathbf{p} S(\mathbf{p})$, then $\mathtt{J}_\mathbf{p}^\top \Lambda_\mathbf{f} \mathtt{J}_\mathbf{p}$ has full rank and we can determine a full distribution for $\mathbf{p}$. A first-order approximation of the covariance $\Sigma_\mathbf{p}$ can be calculated using covariance back-propagation:

$$\Sigma_\mathbf{p} = (\mathtt{J}_\mathbf{p}^\top \Lambda_\mathbf{f} \mathtt{J}_\mathbf{p})^{-1} \ . \tag{8}$$

Note that if $\mathbf{p}$ is high-dimensional, this calculation is very expensive because of the large matrix inversion required.

C. Optimising Rigid Body Transformations in $\mathbb{R}^3$

In optimisation, especially in 3D SLAM, the correct treatment of angles consistently causes confusion. On one hand, a minimal parametrisation is desired, but also singularities should be avoided. Interpolation of angles is not straightforward, since the group of rotation is only locally Euclidean.

Probably the most elegant way to represent rigid body transformations in optimisation is based on Lie theory. A rigid body transformation in $\mathbb{R}^n$ can be expressed as an $(n+1) \times (n+1)$ matrix which can be applied to homogeneous position vectors:

$$\mathtt{T} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathtt{0} & 1 \end{bmatrix} \quad \text{with} \quad \mathtt{R} \in \mathrm{SO}(n) \ , \quad \mathbf{t} \in \mathbb{R}^n \ , \tag{9}$$

and $\mathrm{SO}(n)$ being the Lie group of rotation matrices. The rigid body transformations in $\mathbb{R}^n$ form a smooth manifold and therefore a Lie group — the *Special Euclidean* group $\mathrm{SE}(n)$ [11]. The group operator is the matrix multiplication.

A minimal representation of this transformation is defined by the corresponding Lie algebra $\mathfrak{se}(n)$ which is the tangent space of $\mathrm{SE}(n)$ at the identity. In $\mathbb{R}^3$, the algebra elements are 6-vectors $(\omega, \upsilon)^\top$: $\omega = (\omega_1, \omega_2, \omega_3)$ is the axis-angle representation for rotation, and υ is a rotated version of the translation $\mathbf{t}$. Elements of the $\mathfrak{se}(3)$ algebra can be mapped to the $\mathrm{SE}(3)$ group using the exponential mapping $\exp_{\mathrm{SE}(3)}$:

$$\exp_{\mathrm{SE}(3)}(\omega, \upsilon) := \begin{bmatrix} \exp_{\mathrm{SO}(3)}(\omega) & \mathtt{V}\upsilon \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} . \tag{10}$$

Here, $\exp_{\mathrm{SO}(3)}(\omega) = \mathtt{I} + \frac{\sin(\theta)}{\theta}(\omega)_\times + \frac{1-\cos(\theta)}{\theta^2}(\omega)_\times^2$ is the well-known *Rodrigues' formula*, $\mathtt{V} = \mathtt{I} + \frac{1-\cos(\theta)}{\theta^2}(\omega)_\times + \frac{\theta - \sin(\theta)}{\theta^3}(\omega)_\times^2$, $\theta = ||\omega||_2$, and $(\cdot)_\times$ is an operator which maps a 3-vector to its skew-symmetric matrix. Since $\exp_{\mathrm{SE}(3)}$ is surjective, there exists also an inverse relation $\log_{\mathrm{SE}(3)}$.

During optimisation, incremental updates δ are calculated in the tangent space around the identity $\mathfrak{se}(3)$ and mapped back onto the manifold $\mathrm{SE}(3)$, leading to a modified version of (5):

$$\mathtt{T}_{(i+1)} = \exp_{\mathrm{SE}(3)}(\delta) \cdot \mathtt{T}_{(i)} \ . \tag{11}$$

In this way, we avoid singularities — since δ is always close to the identity — while ensuring a minimal representation during optimisation. The Jacobian of (11) is linear in $\mathtt{T}$,

$$\left.\frac{\partial \exp_{\mathrm{SE}(3)}(\delta) \cdot \mathtt{T}}{\partial \delta}\right|_{\delta=0} = \begin{pmatrix} -(\mathbf{r}_1)_\times & \mathtt{0}_{3\times3} \\ -(\mathbf{r}_2)_\times & \mathtt{0}_{3\times3} \\ -(\mathbf{r}_3)_\times & \mathtt{0}_{3\times3} \\ -(\mathbf{t})_\times & \mathtt{I}_{3\times3} \end{pmatrix} , \tag{12}$$

where $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) := \mathtt{R}$. A general Jacobian $\frac{\partial \mathbf{f}(\exp_{\mathrm{SE}(3)}(\delta) \cdot \mathtt{T})}{\partial \delta}$ can be easily determined using the chain rule.

III. A Keyframe Optimisation System for Large Scale Monocular SLAM

We will now describe the elements of our complete system for sequential monocular SLAM. Similar to Klein and Murray's PTAM system [15], our SLAM framework consists of two parts: a front-end which tracks features in the image and estimates the pose of each live frame given a set of known 3D points, and a back-end which performs joint bundle adjustment over a set of keyframes. In PTAM which focuses on accurate and fast mapping in a small environment, the front-end only performs pose and feature tracking, while the optimisation back-end does everything else such as feature initialisation and dropping of new keyframes. Since our approach focusses on mapping large scenes, we use the opposite emphasis. The optimisation back-end does only bundle adjustment, but the tracking front-end takes over all important decisions such as when to drop new keyframes and which feature to initialise. This design puts a higher computational burden on the tracking front-end, but facilitates exploration since the stability of the tracking front-end does not depend on the optimisation back-end to drop new keyframes.

A. The Camera Projection Function and Camera Poses

Points in the world $\mathbf{x}_j \in \mathbb{R}^3$ are mapped to the camera image using the observation function $\hat{\mathbf{z}}$:

$$\hat{\mathbf{z}}(\mathtt{T}_i, \mathbf{x}_j) = \mathbf{proj}(\mathtt{K} \cdot \mathtt{T}_i \cdot \mathbf{x}_j) \ . \tag{13}$$

Here, $\mathbf{x}_j$ is a homogeneous point, $\mathtt{T}_i \in \mathrm{SE}(3)$, $\mathtt{K}$ is the camera calibration matrix (which we assume is known from prior calibration) and $\mathbf{proj}$ is the 3D-2D projection:

$$\mathbf{proj}(\mathbf{a}) := \frac{1}{a_3}(a_1, a_2)^\top \quad \text{for} \quad \mathbf{a} \in \mathbb{R}^3 \ . \tag{14}$$

We represent the pose of the camera at a given time-step i by the rigid body transformation $\mathtt{T}_i$. Note that the position of the camera center is *not* explicitly represented, but can be recovered as $-\mathtt{R}_i^\top \mathbf{t}_i$ [13, pp.155].

B. Optimisation Back-end

In the back-end, we perform an optimisation jointly over a set of keyframes $\mathtt{T}_{\text{key}:i}$ and large set of points $\mathbf{x}_j$ called bundle adjustment (BA) [29]. In order to be constant-time, the number of keyframes included in BA is restricted. Since we focus on exploration, we optimise over a sliding window of keyframes. At loop closure, different measures have to be taken (see Sec. IV).

In general in BA, not all points are visible in every frame. Let $\mathtt{Z}$ be a sparse matrix where an entry is either an observation $\mathbf{z}_{i,j}$ if point $\mathbf{x}_j$ is visible in frame $\mathtt{T}_{\text{key}:i}$ or zero otherwise. Let I be the sequence of non-zero row indices of $\mathtt{Z}$: $\mathtt{I} = \{(i) : \mathtt{Z}_{i,j} \neq 0\}$. The column index list J is defined similarly. Now, it is obvious how to define the *stacked projection function*:

$$\hat{\mathbf{z}}(\mathtt{T}_{\text{key}:1}, ..., \mathtt{T}_{\text{key}:m}, \mathbf{x}_1, ..., \mathbf{x}_n, \mathtt{I}, \mathtt{J}) \ . \tag{15}$$

We solve BA conventionally using LM by minimising

$$S(\mathbf{p}) = \sum_{(i,j) \in I \times J} (\mathbf{z}_{i,j} - \hat{\mathbf{z}}(\mathtt{T}_{\text{key}:1}, ..., \mathtt{T}_{\text{key}:m}, \mathbf{x}_1, ..., \mathbf{x}_n, \mathtt{I}, \mathtt{J}))^2 \tag{16}$$

with respect to the parameter vector $\mathbf{p} = (\mathtt{T}_{\text{key}:3}, ..\mathtt{T}_{\text{key}:m}, \mathbf{x}_1, ..., \mathbf{x}_n)$. As a sliding window BA is applied, the optimisation window has to be anchored to the previous map. Thus, at least 7 DoF (rotation, translation and scale) should be fixed. Because of this, $\mathtt{T}_{\text{key}:1}, \mathtt{T}_{\text{key}:2}$ are excluded from the optimisation. In order to guard against spurious matches, we use the pseudo-Huber cost function [13, p.619] as a robust kernel.

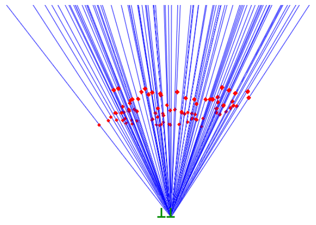
The primary sparseness pattern in BA — that there are only links between points and frames, but no point-point or frame-frame constraints — is exploited in the usual way using the Schur complement [29]. However, our BA implementation also exploits the secondary sparseness pattern — that not every point is visible in every frame. This is done using the CSparse library [6] which performs efficient symbolic Cholesky decomposition using elimination trees.

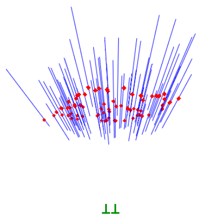
C. Feature Tracking and Pose Optimisation

The main task of the tracking front-end is to estimate the current camera pose $\mathtt{T}_i$. However, it also deals with feature and keyframe initialisation. It takes as input a set of keyframe poses $\mathtt{T}_{\text{key}:i}$ and 3D points $\mathbf{x}_j \in \mathcal{X}$ which are associated with 2D measurements $\mathbf{z}_{\text{key}:i,j}$. The poses $\mathtt{T}_{\text{key}:i}$ and points $\mathbf{x}_j$ were already jointly estimated in the optimisation back-end (see Sec. III-B).

For feature tracking and pose optimisation, we use a mixture of bottom-up (for every pixel) and top-down approaches (guided techniques). First, we estimate the relative pose $\Delta\mathtt{T}_i = \mathtt{T}_{i-1} \cdot \mathtt{T}_i$ between the previous and current frames. This is done by the help of dense variational optical flow implemented very efficiently on the GPU [24]. Given the pose estimate of the previous frame $\mathtt{T}_{i-1}$, we can calculate an initial pose estimate for the current frame: $\mathtt{T}_i^{[0]} = \Delta\mathtt{T}_i \cdot \mathtt{T}_{i-1}^{-1}$. Feature tracking is not done using frame to frame matching. Whereas frame to frame matching is very reliable for pure visual odometry, it prevents mini-loop closures.

Given our initial pose guess $\mathtt{T}_i^{[0]}$, we estimate the subset $\mathcal{X}' \subset \mathcal{X}$ of landmarks which might be visible in the current frame. Based on the projection (13), we perform active search [7] for each point $\mathbf{x}_{j'} \in \mathcal{X}'$, where the centre of the search region is defined by the prediction $\hat{\mathbf{z}}(\mathtt{T}_i^{[0]}, \mathbf{x}_{j'})$. Generally, the uncertainty of the projection and thus the size and shape of the search region depends on the uncertainty of $\mathbf{x}_{j'}$ and $\mathtt{T}_i$. Given that all points in $\mathcal{X}$ are already optimised using keyframe bundle adjustment, and that pose $\mathtt{T}_{i-1}$ is already optimised with respect to the $\mathcal{X}$, then apart from the measurement noise $\Sigma_{\mathbf{z}}$, the only crucial uncertainty originates from the motion between the previous and the current frame. Let, $\Sigma_{\Delta\mathtt{T}_i}$ be the uncertainty of the relative motion. Then the





a) Feature initialisation a) After a single update

Fig. 1. Illustration of the feature initialisation process. First features are initialised as inverse depth points with infinite uncertainty on the depth (a). A single update leads to a significant reduction of the depth uncertainty (b). The inverse depth features are plotted using a 99.7 percent confidence interval.

search area is constrained by the innovation covariance

$$\left(\frac{\partial\hat{\mathbf{z}}(\mathtt{T}',\mathbf{x}_{j'})}{\partial\delta}\right)^\top \Sigma_{\Delta\mathtt{T}_i}\left(\frac{\partial\hat{\mathbf{z}}(\mathtt{T}',\mathbf{x}_{j'})}{\partial\delta}\right)+\Sigma_{\mathbf{z}}\ , \qquad (17)$$

with $\mathtt{T}' = \exp_{\mathrm{SE}(3)}(\delta)\cdot\mathtt{T}_i^{[0]}$. The search within this region is based on template matching using normalised cross-correlation. Using the initial guess $\mathtt{T}_i^{[0]}$, the templates are perspectively warped. In order to speed-up the search, we do not apply the matching at every single pixel in the search region, but only there where FAST features [25] are detected.

As a result, we receive a set of 2D-3D matches between image locations $\mathbf{z}_k$ and feature points $\mathbf{x}_k$. Finally, we optimize the pose $\mathtt{T}_i$ using motion-only BA. Thus, we minimize $S(\mathtt{T}_i) = \sum_k(\mathbf{z}_k - \hat{\mathbf{z}}_k(\mathtt{T}_i,\mathbf{x}_j))^2$ wrt. $\mathtt{T}_i$ using LM. Again, we use the pseudo-Huber cost function as a robust kernel.

D. Feature Initialisation

In monocular SLAM approaches using filtering, no special treatment for feature initialisation is needed if an inverse depth representation [19] is used. New features are jointly estimated together with all other parameters, but at a cost of $O(n^3)$ where n is the number of features visible.

In PTAM [15], new features are triangulated between keyframes. In order to restrict the matching, a strong depth prior is used (restricting the PTAM to using features relatively close to the camera). We suggest a feature initialisation method for keyframe-based SLAM systems based on a set of three dimensional information filters which can estimate the position of arbitrarily distant features. Very recently, a similar method was briefly described by Klein and Murray [14]. Compared to our approach, their filters are applied on every frame, are one-dimensional and are only used for data association.

Ultimately, we would like to update a set of partially initialised 3D points $\mathbf{x}_{new:j}$ given the current camera pose $\mathtt{T}_i$. If $\mathtt{T}_i$ is known, the features $\mathbf{x}_{new:n}$ become independent:

$$p(\mathbf{x}_{new:1},...,\mathbf{x}_{new:j},..|\mathtt{T}_i) = p(\mathbf{x}_{new:1})\cdots p(\mathbf{x}_{new:n})\ . \quad (18)$$

Since the current camera $\mathtt{T}_i$ is well-optimised wrt. the set of known points $\mathcal{X}$ (as described in the previous section), equation (18) is approximately true. Thus, our method employs a set of information filters. Each filter estimates the position of the a single landmark given the current pose estimate. In this sense, our approach is similar to FastSLAM [18]. The difference to FastSLAM is that the partially initialised features $\mathbf{x}_{new:j}$ are not used for state estimation immediately. New features are only used for pose estimation after they are jointly bundle adjusted and added to $\mathcal{X}$ (see Sec. III-E).

The design of a single information filter is greatly inspired by Eade's filtering framework [8]. Features are represented using inverse depth coordinates $\mathbf{y} = (u,v,q)^\top$ wrt. the origin frame $\mathtt{T}_{\mathrm{org}}$ in which they were seen first. They can be mapped to a Euclidean point $\mathbf{x}_{\mathrm{org}} = \frac{1}{q}(u,v,1)^\top$ in this local coordinate frame $\mathtt{T}_{\mathrm{org}}$. The uncertainty of an inverse depth feature is represented using the inverse covariance matrix $\Lambda_{\mathbf{y}} = \Sigma_{\mathbf{y}}^{-1}$. New features are jointly initialised at keyframes where FAST [25] produces candidate locations. In order to ensure an equal distribution of feature locations over the image, we employ a quadtree similar to Mei *et al.* [17]. Given the FAST feature location $\mathbf{z} = (u,v)^\top$, we set $\mathbf{y} = (u,v,q)$ with $q \in \mathbb{R}^+$. The uncertainty $\Lambda_{\mathbf{x}^*}^{(0)}$ is set to $\mathrm{diag}(\frac{1}{\sigma_{\mathbf{z}}^2},\frac{1}{\sigma_{\mathbf{z}}^2},0)$. Note that initially there is an infinite uncertainty along the feature depth, so we do not enforce any depth prior no matter which start value we assign for q. In order to save computation and to avoid divergence, inverse depth features are only updated on selected frames with sufficient parallax.

We filter $\mathbf{y}$ by minimising:

$$S(\mathbf{y}) = (\mathbf{y}-\hat{\mathbf{y}})^\top\Lambda_{\mathbf{y}}(\mathbf{y}-\hat{\mathbf{y}}) + (\Delta\mathbf{z})^\top\Lambda_{\mathbf{z}}(\Delta\mathbf{z}) \qquad (19)$$

wrt. $\mathbf{y}$ using LM. Here, $\hat{\mathbf{y}}$ is set to the initial guess $\mathbf{y}_{[0]}$ and $\Delta\mathbf{z} = \mathbf{z} - \hat{\mathbf{z}}(\mathtt{T}_i, \mathtt{T}_{\mathrm{org}}^{-1}\mathbf{x}_{\mathrm{org}})$. The first term in $S(\mathbf{y})$ ensures that the optimisation of $\mathbf{y}$ is based on its prior distribution $\langle\hat{\mathbf{y}},\Lambda_{\mathbf{y}}\rangle$. The second term takes care that the projection error wrt. the current frame is reduced. In other words, the point $\mathbf{y}$ is moved along its uncertain depth rather than via its certain u,v coordinate in order to reduce the projection error in the current image. We also update the uncertainty using covariance back-propagation (see Fig. 1):

$$\Lambda_{\mathbf{y}}^{(k+1)} = \Lambda_{\mathbf{y}}^{(k)} + \left(\frac{\partial\Delta\mathbf{z}}{\partial\mathbf{y}}\right)^\top\Lambda_{\mathbf{z}}^{(k)}\left(\frac{\partial\Delta\mathbf{z}}{\partial\mathbf{y}}\right)\ . \qquad (20)$$

E. Keyframe Dropping

A new keyframe is dropped as soon the distance of the current camera from the closest keyframe exceeds a threshold. Only inverse depth features $\mathbf{y}$ which could be reliably tracked and whose position is certain enough are considered for adding to $\mathcal{X}$. Before, they are bundle adjusted with respect to a small number (e.g. three) of surrounding keyframes. Only those features which do not produce a large projection error in any of the surrounding keyframes are added to $\mathcal{X}$.

IV. Loop Closure

A. Loop closure detection

It is well known that loop closures can be detected effectively using appearance information only [1, 5]. These methods often rely on visual bags of words based on SIFT or SURF features. Given that we have a loop closure detection between two frames each associated with a set of SURF features, the standard method would apply RANSAC in conjunction with the 5-point method [21] in order to estimate the epipolar geometry. Then the relative Euclidean motion up to an unknown scale in translation can be recovered.

However, we can exploit the fact that in our SLAM system each frame is associated with a large set of three-dimensional feature points. First, we create a candidate set of SURF feature pairs by matching features between the current frame and the loop frame based on their descriptors. Then, we create a dense surface model using the three-dimensional feature points visible in the loop frame. Next, the unknown depth of the candidate SURF features of the loop frame is calculated using this dense surface model.[1] The computationally very efficient k-nearest neighbour regression algorithm proved to be sufficient for our needs. In other words, we calculate the depth of SURF features by simply interpolating the depth of nearby 3D points.

Finally, a 6 DoF loop constraint $\mathtt{T}_{\text{loop}}$ is calculated based on the 2D-3D SURF correspondences using the P4P-method plus RANSAC. Given this initial pose estimate, more matches can be found using perspective-warped image patches, and the pose is refined using robust optimisation. However the relative change in scale s_{loop} due to scale drift has to be computed as well, so we also calculate the depth of the SURF features visible in the current frame using a dense surface model, giving a set of 3D-3D correspondences. We estimate s_{loop} robustly as the median scale difference over all correspondences.

B. Loop Closure Correction

Let us consider a loop closure scenario in a large-scale map. After the loop closure is detected and matches are found between the current frame and the loop frame, the loop closing problem can be stated as a large BA problem. We have to optimize over all frames and points in the map. However, optimising over a large number of frames is computationally demanding. More seriously, since BA is not a convex problem, and we are far away from the global minimum, it is likely that BA will get stuck in a local minimum.

One solution would be to optimise over relative constraints between poses using pose-graph optimisation [12][22]. The relative constraints between two poses $\mathtt{T}_i$ and $\mathtt{T}_j$ can be calculated easily: $\Delta\mathtt{T}_{i,j} = \mathtt{T}_j \cdot \mathtt{T}_i^{-1}$. But while an optimisation over the 6 DoF constraints would efficiently correct translational and rotational drift, it would not deal with scale drift, and would lead to an unsatisfactory overall result as we will show experimentally in Section V. Therefore, we perform optimisation based on 7 DoF constraints $\mathtt{S}$, which are elements of the group of similarity transformations Sim(3):

$$\mathtt{S} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{21}$$

[1]The underlying assumption is that the scene structure is smooth enough. However, this assumption is not only vital if dense surface models are used, but always if we aim to calculate the 3D position of a blob feature no matter which method is used. In other words, the 3D position of a blob is only properly defined if its 'carrying' surface is smooth enough.

where $s \in \mathbb{R}^+$ is a positive scale factor. A minimal representation $\mathfrak{sim}(3)$ can be defined be the following 7-vector $(\omega, \sigma, \upsilon)^\top$ with $s = e^\sigma$. Indeed, it can be shown that Sim(3) is a Lie group, $\mathfrak{sim}(3)$ is the corresponding Lie algebra and their exponential map is:

$$\exp_{\mathrm{Sim}(3)}\begin{pmatrix} \omega \\ \sigma \\ \upsilon \end{pmatrix} = \begin{bmatrix} e^\sigma \exp_{\mathrm{SO}(3)}(\omega) & \mathtt{W}\upsilon \\ \mathtt{0} & 1 \end{bmatrix} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{22}$$

with $\mathtt{W} = e^\sigma\left(\mathtt{I} + \frac{1-\cos(\theta)}{\theta^2}(\omega)_\times + \frac{\theta-\sin(\theta)}{\theta^3}(\omega)_\times^2\right)$. Furthermore, $\exp_{\mathrm{Sim}}(3)$ is surjective, so an inverse relation $\log_{\mathrm{Sim}(3)}$ exists. Similarly to SE(3), the Jacobian of the incremental update $\exp_{\mathrm{Sim}(3)}(\delta) \cdot \mathtt{S}$ is linear in $\mathtt{S}$.

In order to prepare for the 7 DoF optimisation, we transform each absolute pose $\mathtt{T}_i$ to an absolute similarity $\mathtt{S}_i$, and each relative pose constraint $\Delta\mathtt{T}_{i,j}$ to a relative similarity constraint $\Delta\mathtt{S}_{i,j}$ by leaving rotation and translation unchanged and setting the scale $s = 1$. Only for the loop constraint $\Delta\mathtt{S}_{\text{loop}}$ do we set the scale to $s_{\text{loop}} \neq 1$ (using the estimate obtained as explained in the previous section). We define the residual $\mathbf{r}_{i,j}$ between two transformations $\mathtt{S}_i$ and $\mathtt{S}_j$ minimally in the tangent space $\mathfrak{sim}(3)$:

$$\mathbf{r}_{i,j} = \log_{\mathrm{Sim}(3)}(\Delta\mathtt{S}_{i,j} \cdot \mathtt{S}_i \cdot \mathtt{S}_j^{-1}) \ . \tag{23}$$

The graph of similarity constraints is optimised by minimising:

$$S(\mathtt{S}_2, ..., \mathtt{S}_m) = \sum_{i,j} \mathbf{r}_{i,j}^\top \Lambda_{\Delta\mathtt{S}i,j} \mathbf{r}_{i,j} \tag{24}$$

using LM. The first transform $\mathtt{S}_1$ is fixed and defines the coordinate frame. The corresponding Jacobian is sparse. We exploit the sparseness pattern using the CSparse library.

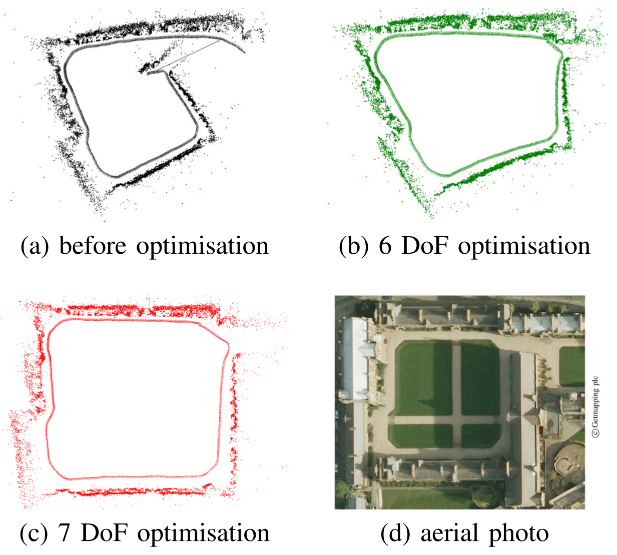
After the similarities $\mathtt{S}_i^{\text{cor}}$ are corrected, we also need to correct the set of points. For each point $\mathbf{x}_j$, a frame $\mathtt{T}_i$ is selected in which it is visible. Now we can map each point relative to its corrected frame:

$$\mathbf{x}_j^{\text{cor}} = (\mathtt{S}_i^{\text{cor}})^{-1}(\mathtt{T}_i\mathbf{x}_j) \ . \tag{25}$$

Afterwards, each similarity transform $\mathtt{S}_i^{\text{cor}}$ is transformed back to a rigid body transform $\mathtt{T}_i^{\text{cor}}$ by setting the translation to $s\mathbf{t}$ and leaving the rotation unchanged. Finally, the whole map can be further optimised using structure-only or full BA.

V. Experiments

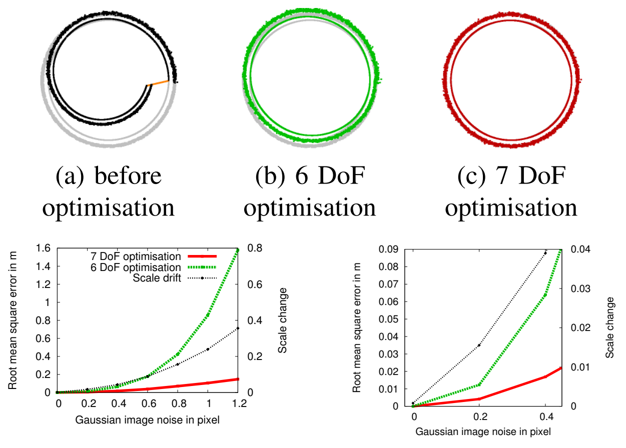
We performed the evaluation of our SLAM system using the Keble College data set [4] — a sequence where a hand-held sideways-facing camera completes a circuit around a large outdoor square. Images were captured using a low cost IEEE Unibrain camera with resolution 320×240 at 30 FPS, and using a lens with 80 degree horizontal field of view. Our SLAM framework performed almost in real-time. For the 5952 images in the sequence, the run-time was 465 seconds which corresponds to frame rate of more than 12 FPS. The computation was performed on a desktop computer with an Intel Core 2 Duo processor and an NVIDIA 8800 GT GPU which were used for dense optical flow computation.



(a) before optimisation (b) 6 DoF optimisation

(c) 7 DoF optimisation (d) aerial photo

Fig. 2. Keble college data set



(a) before optimisation (b) 6 DoF optimisation (c) 7 DoF optimisation

(d) whole plot (e) close-up

Fig. 3. Simulation experiments

The trajectory before loop closure is shown in Fig. 2 (a). It consists of 766 keyframes, 11885 points and 84820 observations. In the optimisation back-end, BA with a sliding window of 10 keyframes was used. A loop closure constraint $\Delta\mathtt{T}_{\text{loop}}$ was successfully calculated and refined using the method described in Sec.IV. A relative scale change of $1 : 0.37$ was detected. The large amount of drift can be partially explained by the fact that the visible scene is always very local and that only a rough intrinsic calibration was available. Also, future improvements in our visual odometry implementation could lead to a reduction of drift during exploration. Nevertheless, a certain amount of drift during exploration is unavoidable and our main focus is how to deal with drift when it occurs. A traditional 6 DoF graph optimisation such as [12] closes the loop but leaves the scale drift unchanged which leads to a deformed trajectory as shown in Fig. 2 (b). However, if we perform graph optimisation using the similarity transform, the result looks significantly better (see Fig. 2 (c)). The graph optimisation converged after two iterations, taking 1.05 seconds in total. After graph optimisation, ten iterations of structure-only bundle adjustment were performed, taking 0.31 seconds. The application of full BA over all 766 keyframes did not lead to a notable further improvement (not shown here), but 10 iterations would take over 200 seconds.

In addition to this real-world experiment, we also performed a series of simulation experiments in full 3D space. A simulated camera was moved on a circular trajectory with radius 10m. The camera is directed outwards. A set of 5000 points was drawn from a ring shaped distribution with radius 11m. The camera trajectory consists of 720 poses. In this simulation environment, our full SLAM system was applied including feature initialisation and fixed-window bundle adjustment with size 10. Only the camera was not simulated and data association was given. The difference between the true trajectory and the estimated one is shown in Fig. 3 (a). In this particular example, we simulated Gaussian image noise with a standard deviation of one pixel. Fig. 3 (b) and (c) show loop closure correction using $\mathrm{SE}(3)$ and $\mathrm{Sim}(3)$.

We varied the amount of image noise from 0 to 1.2 pixels and performed 10 simulation runs for each setting. An important result of our experiment is that there is a clear relation between scale drift and image noise (see Fig. 3 (d), thin curve). This indicates the correctness of our characterisation of scale as a parameter in SLAM which drifts during exploration in a similar way to rotation and translation. If we define the first pose as our origin, there is still a scale ambiguity of possible maps. Therefore we define a measure between the corrected poses $\mathtt{T}_i^{\text{cor}}$ and the true poses $\mathtt{T}_i^{\text{true}}$ using the minimum of the sum of square differences

$$M = \min_s \sum_i (\mathbf{t}_i^{\text{true}} - s\mathbf{t}_i^{\text{cor}})^2 \qquad (26)$$

over the scale s. By dividing M by the number of frames and taking the square root, we obtain the root mean square error $RMSE = \sqrt{\frac{M}{720}}$. The average RMSE over the ten simulation runs is shown in Fig. 3 (d). One can see that $\mathrm{Sim}(3)$ optimisation (red curve) outperforms $\mathrm{SE}(3)$ optimisation (green curve) by a large amount, particularly if the scale change is large. But interestingly, even for small scale changes of one to four percent the $\mathrm{Sim}(3)$ optimisation performs significantly better than $\mathrm{SE}(3)$ optimisation (see Fig. 3 (e)).

Finally we did an experiment to illustrate that our optimisation framework naturally extends to multiple loop closures. Imagine an aircraft flying over a sphere and performing monocular SLAM using a downward directed camera (see Fig. 4 (a)). First we perform visual odometry, and afterwards we compute a set of loop closure constraints (shown as blue line segments in Fig. 4 (b)). In this particular example, $\mathrm{Sim}(3)$ optimisation leads to a small RMSE of only 0.328 (Fig. 4 (c)), whereas $\mathrm{SE}(3)$ optimisation results in an RMSE of 2.187. Videos illustrating the experiments are available on-line[2].

[2]`http://www.doc.ic.ac.uk/~strasdat/rss2010videos`

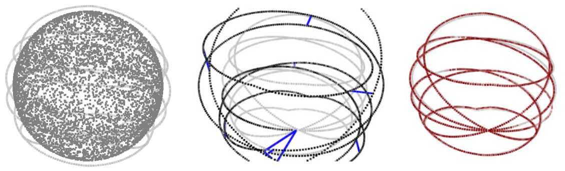


Fig. 4. Multi-loop-closure example: An aircraft is flying over a sphere and doing SLAM using a single downward-directed camera.

VI. Conclusions

We have presented a new monocular SLAM algorithm, based consistently on non-linear optimisation, which we believe takes advantage of the best current knowledge of the hard problem of large scale monocular SLAM to achieve state of the art results. Our algorithm is suitable for sequential operation in large domains, currently at near real-time rates. Importantly, our approach explicitly acknowledges the issue of scale drift at all stages, and offers a practical way to resolve this drift effectively upon loop closures. We show experimentally, both in simulation and using a real outdoor sequence, that a certain amount of scale drift is unavoidable during exploration and this must be taken into account during loop closure to achieve the best results. Also, we have shown that our optimisation approach naturally extends to multiple loop closures.

BA over all frames and scene points is the gold standard for SfM problems where camera networks are typically strong. However, a lightweight approach of 7 DoF graph-optimisation plus structure-only BA seems to be sufficient to accurately close large loops. It is unclear and worth further investigating whether full BA leads to significant better results for the weak camera networks which occur in monocular SLAM.

Acknowledgments

This research was supported by the European Research Council Starting Grant 210346, the Spanish MEC Grant DPI2009-07130 and EU FP7-ICT-248942 RoboEarth. We are very grateful to colleagues at Imperial College London and the University of Zaragoza for discussions and software collaboration. Special thanks to Giorgio Grisetti for the insightful discussion about pose-graph optimisation.

References

[1] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics (T-RO)*, 24(5):1027–1037, 2008.

[2] D. Chekhlov, M. Pupilli, W. W. Mayol, and A. Calway. Real-time and robust monocular slam using predictive multi-resolution descriptors. In *Proceedings of the 2nd International Symposium on Visual Computing*, 2006.

[3] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point RANSAC for EKF-based structure from motion. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2009.

[4] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems (RSS)*, 2007.

[5] M. Cummins and P. Newman. Highly scalable appearance-only SLAM — FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[6] T. A: Davis. *Direct Methods for Sparse Linear Systems*. SIAM, 2006.

[7] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[8] E. Eade. *Monocular Simultaneous Localisation and Mapping*. PhD thesis, University of Cambridge, 2008.

[9] E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[10] E. Eade and T. Drummond. Monocular SLAM as a graph of coalesced observations. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007.

[11] J. Gallier. *Geometric Methods and Applications for Computer Science and Engineering*. Springer-Verlag, 2001.

[12] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, 2010.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[14] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, Orlando, October 2009.

[15] G. Klein and D. W. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[16] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics (T-RO)*, 24:1066–1077, 2008.

[17] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A constant time efficient stereo SLAM system. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[18] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *IEEE International Conference on Robotics and Automation*, 2003.

[19] J. M. M. Montiel, J. Civera, and A. J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2006.

[20] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real-time localization and 3D reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[21] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, 2004.

[22] E. Olson, J. J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[23] P. Pinies and J. D. Tardós. Large scale SLAM building conditionally independent local maps: Application to monocular vision. *IEEE Transactions on Robotics (T-RO)*, 24(5):1094–1106, 2008.

[24] T. Pock, M. Unger, D. Cremers, and H. Bischof. Fast and exact solution of total variation models on the GPU. In *Proceedings of the CVPR Workshop on Visual Computer Vision on GPU's*, 2008.

[25] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2005.

[26] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[27] J. Solà, M. Devy, A. Monin, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2005.

[28] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular SLAM: Why filter? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[29] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment — a modern synthesis. In *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.