2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Real-time Monocular SLAM: Why Filter?

Hauke Strasdat, J. M. M. Montiel and Andrew J. Davison

Abstract— **While the most accurate solution to off-line structure from motion (SFM) problems is undoubtedly to extract as much correspondence information as possible and perform global optimisation, sequential methods suitable for live video streams must approximate this to fit within fixed computational bounds. Two quite different approaches to real-time SFM — also called monocular SLAM (Simultaneous Localisation and Mapping) — have proven successful, but they sparsify the problem in different ways. Filtering methods marginalise out past poses and summarise the information gained over time with a probability distribution. Keyframe methods retain the optimisation approach of global bundle adjustment, but computationally must select only a small number of past frames to process.**

In this paper we perform the first rigorous analysis of the relative advantages of filtering and sparse optimisation for sequential monocular SLAM. A series of experiments in simulation as well using a real image SLAM system were performed by means of covariance propagation and Monte Carlo methods, and comparisons made using a combined cost/accuracy measure. With some well-discussed reservations, we conclude that while filtering may have a niche in systems with low processing resources, in most modern applications keyframe optimisation gives the most accuracy per unit of computing time.

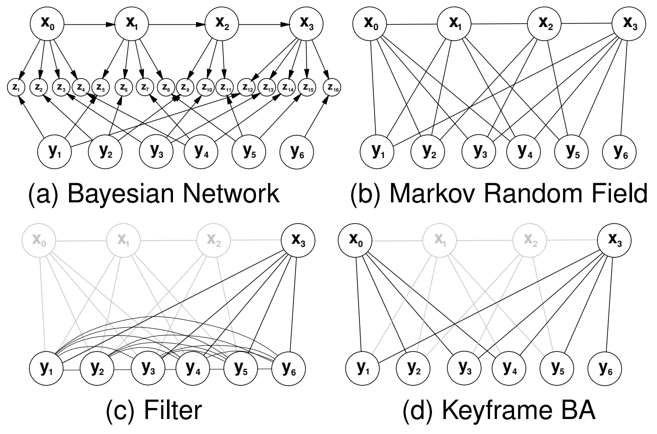


Fig. 1. (a) Bayesian network for SLAM/SFM. (b) SLAM/SFM as markov random field without representing the measurements explicitly. (c) and (d) visualise how inference progressed in a filter and with keyframe-based optimisation.

I. Introduction

Live motion and structure estimation from a single moving video camera has potential applications in domains such as robotics, wearable computing, augmented reality and the automotive sector. This research area has a long history dating back to work such as [10], but the past five years — through advances in computer processing power as well as algorithms — have seen great progress and several standout demonstration systems have been presented. Two methodologies have been prevalent: filtering approaches (e.g. [6], [8]) which fuse measurements from all images sequentially by updating probability distributions over features and camera pose parameters; and bundle adjustment style optimisation over selected images from the live stream, such as a sliding window [18], [20], or in particular spatially distributed keyframes [13], [14] which permit drift-free long-term operation.

Understanding of the generic character of localisation and reconstruction problems has recently matured significantly. In particular, recently a gap has been bridged between the Structure from Motion (SFM) research area in computer vision, whose principles were derived from photogrammetry, and the Simultaneous Localisation and Mapping (SLAM) sub-field of mobile robotics research — hence the somewhat unfortunate dual terminology. The essential character of these two problems, estimating sensor motion by modelling the previously unknown but static environment, is the same, but the motivation of researchers has historically been different. SFM tackled problems of 3D scene reconstruction from small sets of images, and projective geometry and optimisation have been the prevalent methods of solution. In SLAM, on the other hand, the classic problem is to estimate the motion of a moving robot in real-time as it continuously observes and maps its unknown environment with sensors which may or may not include cameras. Here sequential filtering techniques have been to the fore.

It has taken the full adoption of Bayesian methods for both to be able to be understood with a unified single language and a full cross-over of methodologies to occur. Recent papers such as [12], [15], [17] now pull together the best of both approaches. There remains, however, the fact that in the specific problem of real-time monocular camera tracking, the best systems have been strongly tied to one approach or the other. The question of why, and whether one approach is clearly superior to the other, needs resolving to guide future research in this important application area.

In this paper we compare the filtering and optimisation approaches to monocular SLAM, analysing both for the first time in terms of how they trade off accuracy for computation speed via comprehensive simulation experiments and

This research was supported by the European Research Council Starting Grant 210346, the Spanish MEC Grant DPI2009-07130 and EU FP7-ICT-248942 RoboEarth.

H. Strasdat and A. J. Davison are with the Department of Computing, Imperial College London, UK. {strasdat,ajd}@doc.ic.ac.uk
J.M.M. Montiel is with the Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Spain. josemari@unizar.es

representative experiments with real images, and reach some notable conclusions. The following section discusses the problem in general terms, then we propose an experimental basis for the analysis and proceed to results and discussion.

II. Filtering versus Optimisation

The general problem of SLAM/SFM can be posed in terms of inference on a graph. Initially, we represent the variables involved by the Bayesian network shown in Figure 1(a). The variables of interest are $\mathbf{x}_i$, each a vector of parameters representing a historic position of the camera, and $\mathbf{y}_i$, each a vector of parameters representing the position of a feature, assumed to be static. These are linked by image feature measurements $\mathbf{z}_i$, each depending on one feature and one pose, and sometimes links between consecutive $\mathbf{x}_i$ representing non-visual knowledge about local motion, for instance from odometry or smoothness assumptions. In real-time SLAM, this network will continuously grow as new pose and measurement variables are added at every time step, and new feature variables will be added whenever new parts of a scene are explored for the first time.

In Fig. 1(b), the poses $\mathbf{x}_i$ and features $\mathbf{y}_i$ are presented in a markov random field without making the known conditionals $\mathbf{z}_i$ explicit. Although various parametric and non-parametric inference techniques have been applied to SFM and SLAM problems (such as particle filters, or global optimisation based on the L_∞ norm), the most generally successful methods in both filtering and optimisation have assumed Gaussian distributions for measurements and ultimately state-space estimation; equivalently we could say that they are least-squares methods in the reprojection error minimised. Standard bundle adjustment (BA) in SFM, or the Extended Kalman Filter (EKF) and variants in SLAM all manipulate the same types of matrices representing Gaussian means and covariances. The clear reason is the special status of the Gaussian as the central distribution of probability theory which makes it the most efficient way to represent uncertainty in a wide range of practical inference. We therefore restrict our analysis to this domain.

A direct application of optimal BA to sequential SLAM would involve finding the full maximum likelihood solution to the graph of Fig. 1(b) from scratch as it grew at every new time-step. The computational cost would clearly get larger at every frame, and quickly out of hand. In inference suitable for real-time implementation, we therefore face two key possibilities in order to avoid computational explosion.

In the *filtering* approach illustrated by Fig. 1(c), all poses other than the current one are marginalised out after every frame. Features, which may be measured again in the future, are retained. The result is a graph which stays relatively compact; it will not grow arbitrarily with time, and will not grow at all during repeated movement in a restricted area, adding persistent feature variables only when new areas are explored. The downside is that the graph quickly becomes fully inter-connected, since every elimination of a past pose variable causes fill-in with new links between every pair of feature variables to which it was joined. Joint potentials over all of these mutually-interconnected variables must therefore be stored and updated. The computational cost of propagating joint distributions scales poorly with the number of variables involved, and this is the main drawback of filtering: in SLAM, the number of features in the map will be severely limited. The standard algorithm for filtering using Gaussian probability distributions is the EKF, where the dense inter-connections between features are manifest in a single joint density over features stored by a mean vector and large covariance matrix.

The other option is to retain BA's *optimisation* approach, solving the graph from scratch time after time as is grows, but to sparsify it by removing all but a small subset of past poses. In some applications it is sensible for the retained poses to be in a sliding window of the most recent camera positions, but more generally they are a set of intelligently or heuristically chosen *keyframes* (see Fig. 1(d)). The other poses, and all the measurements connected to them, are not marginalised out as in the filter, but simply discarded — they do not contribute to estimates. Compared to filtering, this approach will produce a graph which has more elements (since many past poses are retained), but importantly for inference the lack of marginalisation means that it will remain sparsely inter-connected. The result is that graph optimisation remains relatively efficient, *even if the number of features in the graph and measured from the keyframes is very high*. The ability to incorporate more feature measurements counters the information lost from the discarded frames.

So the key question is whether it makes sense to summarise the information gained from historic poses and measurements by joint probability distributions in state space and propagate these through time (filtering), or to discard some of those measurements in such a way that repeated optimisation from scratch becomes feasible (keyframes), and propagating a probability distribution through time is unnecessary. We have not seen a convincing investigation of which approach works best in the specific problem of monocular SLAM, and in this paper propose a set of experiments to investigate this issue in detail in terms of the trade-off between accuracy and computational cost.

III. Defining an Experimental Setup

We keep two key state of the art systems in mind in our definition of simulation experiments: Klein and Murray's breakthrough keyframe based SLAM system [13], [14], and Eade and Drummond's system which builds a graph of locally filtered sub-maps [8]. These systems are similar in many regards, incorporating parallel processes to solve local metric mapping, appearance-based loop closure detection and background global map optimisation over a graph. They are very different at the very local level, however, in exactly the way that we wish to investigate, in what constitutes the fundamental building block of their mapping processes. In Klein and Murray's system, it is the keyframe, a historical pose of the camera where a large number of features are matched and measured. *Only information from these keyframes* goes into the final map — all other frames are used locally for tracking but that information is ultimately discarded. Klein and Murray's key observation which permits real-time operation is that BA over keyframes does not have to happen at frame-rate. In their implementation, BA runs in one thread on a multi-core machine, completing as often as possible, while a second tracking thread does operate at frame-rate with the task of pose estimation of the current camera position with respect to the fixed map defined by the nearest keyframe.

In Eade and Drummond's system, the building block is a 'node', which is a filtered probabilistic sub-map of the locations of features. *Measurements from all frames* are digested in this sub-map, but the number of features it contains is consequently much smaller. In Eade and Drummond's system, updates to the local nodes (sub-maps) are carried out using non-linear optimisation rather than an EKF step and they use an inverse covariance (information) representation. However, the important point for our purposes is that both the mean vector of estimated feature locations and a covariance matrix representing uncertainty are updated at every new image and a probability distribution therefore summarises past measurements as in [6]. There would be little difference if the EKF was used locally.

The spacing of Klein and Murray's keyframes and Eade and Drummond's nodes is decided automatically in both cases, but turns out to be similar. Essentially, during a camera motion between two neighbouring keyframes or nodes, a high fraction of features in the image will remain observable. So in our simulations, we aim to isolate this very local part of the general mapping process: the construction of a building block which is one node or the motion between two keyframes (see the camera motions in Fig. 2).

We wish to analyse both accuracy and computational cost. As a measure of accuracy, we consider only the error between the start and end point of a camera motion. This is appropriate as it measures how much camera uncertainty grows with the addition of each building block to a large map. An important aspect of the analysis is that we obtain accuracy measures from a single process as a function of number of features and number of intermediate frames — bundle adjustment — rather than specifically simulating the filtering process. It is well known that inaccuracy can accumulate in filters due to limitations in the representation of probability distributions, and specifically the baked-in linearisation assumptions of Gaussian-based filters like the EKF. Improving the filtering parameterisation (as in [4]) and increasing feature counts can drastically reduce these problems. But the BA accuracy measure we use is therefore an upper bound on what would be achievable using filtering.

IV. Preliminaries

A. Camera Projection Model

The projection model which maps points $\mathbf{y}$ in the world to image coordinates $\hat{\mathbf{z}}$ can be specified as

$$\hat{\mathbf{z}} = \begin{pmatrix} f & 0 & p_1 \\ 0 & f & p_2 \\ 0 & 0 & 1 \end{pmatrix} [\mathtt{R}|\mathbf{t}]\mathbf{y}, \qquad (1)$$

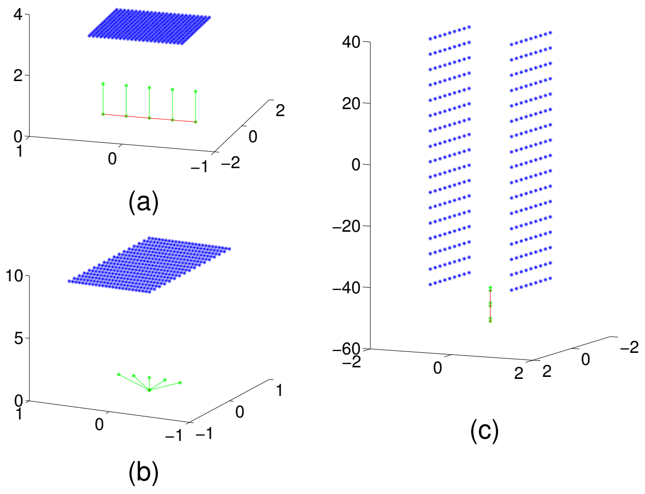


Fig. 2. We analysed three different scene settings: Sideways motion (a), almost pure rotation (b) and forward motion in a corridor environment (c). The camera trajectory is displayed as a (red) curve. The orientation of the camera is highlighted using a (green) vector at intermediate positions. The planar object is represented by a number of (blue) points.

where f is the focal length in terms of pixel dimensions, $(p_1, p_2)^\top$ the principal point, $\mathtt{R}$ is a 3×3 matrix representing the camera rotation, and $\mathbf{t}$ is a 3-vector [11, pp. 154]. Note that the position of the camera centre $\mathbf{C}$ in the world frame is not represented explicitly but can be obtained as follows:

$$\mathbf{C} = -\mathtt{R}^\top \mathbf{t} \ . \qquad (2)$$

V. Analysis of Accuracy and Cost

A. Motion and Structure in the Simulated Scene

In simulation, we consider a pinhole camera with focal length $f = 1000$, a resolution of 1280×960 and a principal point at $(p_1, p_2)^\top = (640, 480)^\top$.

In our standard experiment, we consider the following scene. The camera performs a sideways motion (see Fig. 2 (a)). The camera trajectory is 1m. A bounded fronto-parallel planar object at 3m depth is visible in the scene. The object is fully observable from all intermediate frames. In addition, we also analyse other scenarios such as rotation with very little translation and forward motion in a corridor environment (see Fig. 2 (b) and 2 (c) respectively); generally we have found the results obtained to be very similar.

In the experiments, we vary two essential parameters, the total number of frames and the number of point feature observations. Note that we denote the number of cameras involved in the SLAM estimate by m if the camera trajectory consists of $m+1$ frames. This is because the pose of the first camera defines the coordinate frame and is not estimated. The number of observations per frame is equal to the number of 3D points in the scene since the whole scene is observable from all camera frames. We vary the number of frames in the range $m \in \{1, 2, \ldots 16\}$, and the number of features in the range $n \in \{12, \ldots 425\}$. The scene points are arranged

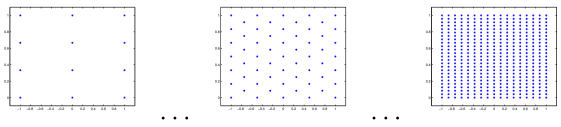


Fig. 3. Regular pattern consisting of 12 to 425 points.

in a regular pattern (see Fig. 3). The number of points is increased such that a pattern with a higher density always includes all points from a pattern with lower density.

B. Accuracy Analysis

In BA, we seek to estimate the state vector $\mathbf{p}$ over the joint state of all camera poses $\mathbf{x}_i$ and scene points $\mathbf{y}_j$:

$$\mathbf{p} = (\mathbf{x}_1^\top, \ldots, \mathbf{x}_m^\top, \mathbf{y}_1^\top, \ldots, \mathbf{y}_n^\top)^\top \ . \tag{3}$$

A camera pose is represented by a rotation matrix $\mathtt{R}$ and translation vector $\mathbf{t}$. It is desirable to have a minimal representation of camera rotation. Let $\mathbf{q}_i$ be a unit quaternion which corresponds to the rotation matrix $\mathtt{R}_i$. We define the rotation of the ith camera frame as a 3-vector $\mathbf{r}_i$ representing the imaginary parts of $\mathbf{q}_i$. Thus, the ith camera pose is fully specified by:

$$\mathbf{x}_i = (\mathbf{r}_i^\top, \mathbf{t}_i^\top)^\top \ . \tag{4}$$

For BA, we employ the well known high-performance SBA library [16]. The sparse Jacobian $\mathtt{J}$ used for the Levenberg-Marquardt algorithm is:

$$\mathtt{J} = \frac{\partial \hat{\mathbf{z}}(\mathbf{p})}{\partial \mathbf{p}} \ . \tag{5}$$

The accuracy of a specific SLAM setting is evaluated using backward propagation of covariance [11, pp. 141]. The measurement noise is set to a standard deviation of $\sigma = 0.5$ pixels:

$$\Sigma_{\mathbf{z}} = \mathrm{diag}(\sigma^2, \ldots, \sigma^2) \ . \tag{6}$$

However, the state vector $\mathbf{p}$ is over-parameterised since in monocular SLAM we can only estimate the joint state of structure and motion modulo an unknown scale factor. Hence, the matrix $\mathtt{J}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J}$ is rank deficient, not invertible, and thus we cannot perform covariance back-propagation straightaway. Therefore, we define a transformation which maps the over-parameterised state $\mathbf{p}$ to the minimal representation $\tilde{\mathbf{p}}$. First, all translation parameters $\mathbf{t}_i$ and structure parameters $\mathbf{y}_j$ are scaled by a factor ρ such that $|\rho \mathbf{t}_m| = 1$. Then, we represent the position of the final translation $\mathbf{t}_m$ using two parameters only — the spherical angles θ_m and ϕ_m. Thus, the final camera pose is represented minimally by:

$$\tilde{\mathbf{x}}_m = (\mathbf{r}_m^\top, \theta_m, \phi_m)^\top \ . \tag{7}$$

Now, we can calculate the covariance of the minimal parameterisation $\Sigma_{\tilde{\mathbf{p}}}$:

$$\Sigma_{\tilde{\mathbf{p}}} = (\mathtt{H}^\top \mathtt{J}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J} \mathtt{H})^{-1} \tag{8}$$

where H is the Jacobian corresponding to the mapping from the minimal state $\tilde{\mathbf{p}}$ to the over-parameterised one $\mathbf{p}$. In the end, we are only interested in the 5×5 sub-matrix $\Sigma_{\tilde{\mathbf{x}}_m}$ which specifies the uncertainty of the final camera pose $\tilde{\mathbf{x}}_m$.

We analyse the influence of different parameters $\langle m, n \rangle$ in terms of entropy reduction. Therefore, for each setting $\langle m, n \rangle$ we compute the covariance matrix of the final camera pose $\Sigma_{\tilde{\mathbf{x}}_m}^{\langle m,n \rangle}$ as described above. Then, we can compute the entropy reduction in bits,

$$\log_2 \left(\frac{\det\left(\Sigma_{\tilde{\mathbf{x}}_m}^{\langle m,n \rangle}\right)}{\det\left(\Sigma_{\tilde{\mathbf{x}}_m}^{\langle 1,12 \rangle}\right)} \right) , \tag{9}$$

in relation to the least accurate case $\langle 1, 12 \rangle$ where only one frame and 12 scene points are used for the optimisation.

The influence of the parameters $\langle m, n \rangle$ for the sideways motion case is illustrated in Fig. 4. As can be seen in all plots, increasing the number of features significantly reduces the entropy. On the other hand, increasing the number of intermediate frames has only a minor influence. This is the single most important result of this paper. For all three different settings — sideways motion, rotation, and forward motion (see Fig. 4(a-c)) — the plot looks very similar. However, in the latter case it seems to be a bit more promising to increase the number of frames.

Since we can only perform experiments for a discrete number of settings $\langle m, n \rangle$, the resulting entropy-reduction surfaces (Fig. 4(a-c)) are represented sparsely. However, for the following analysis it is essential to have a continuous representation. We have found that the surfaces are represented well by the following function:

$$v(n, m) = \alpha_1 \log(n + \alpha_4) + \alpha_2 \log(m + \alpha_5) + \alpha_3 \ . \tag{10}$$

Using non-linear least squares regression, the parameter vector α can be estimated robustly. The resulting surface for the sideways motion case is shown in Fig. 4(d). The function represents the sparse data well, which is reflected by a Normalised Root Means Square Error (NRMSE) of only 0.00769.

For comparison reasons, we also conducted a Monte-Carlo experiment instead of covariance propagation. For each pair $\langle m, n \rangle$, we perform BA 1000 times, sampling measurement noise in each trial from a Gaussian with zero mean and covariance $\Sigma_{\mathbf{z}}$ as defined in (6). Then, the covariance of the final camera pose is computed from the 1000 BA runs. The resulting accuracy surface — again, in terms of entropy reduction — can be seen in Fig. 4 (e).

C. Accuracy Analysis using Real Image Sequences

In addition to the simulation experiments, we performed a series of Monte Carlo experiments using real image sequences (see Fig. 5 (a)). For this we developed a keyframe-based SLAM framework similar to Klein and Murray's system. At each frame, an initial pose is estimated from previously estimated 3D structure using SURF [2] and RANSAC-based Perspective-Four-Point (P4P) camera estimation followed by an iterative refinement. By means of this initial pose estimate, more 2D-3D correspondences are found using warped patch matching. New 3D points are initialised

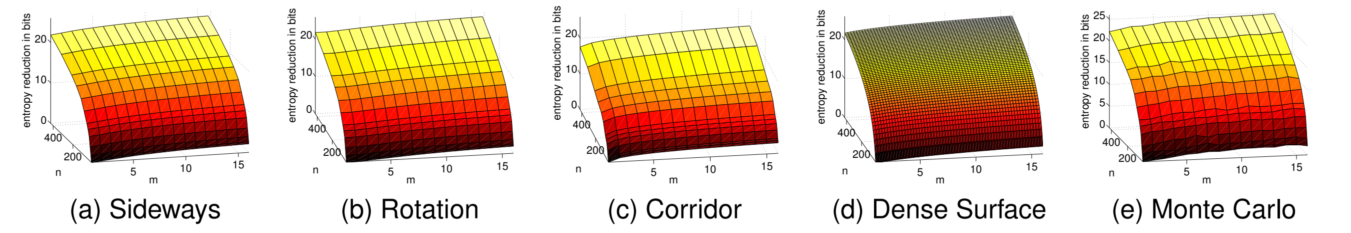


Fig. 4. The influence of the number of features n and the number of frames m on camera motion estimation accuracy in terms of entropy reduction. In (a), (b), and (c), sparse plots for sideways, rotation and forward motions are displayed respectively. (d) shows a denser surface for the sideways experiment fitted by least squares regression. Monte-Carlo experiments lead to comparable results as can be seen in (e).

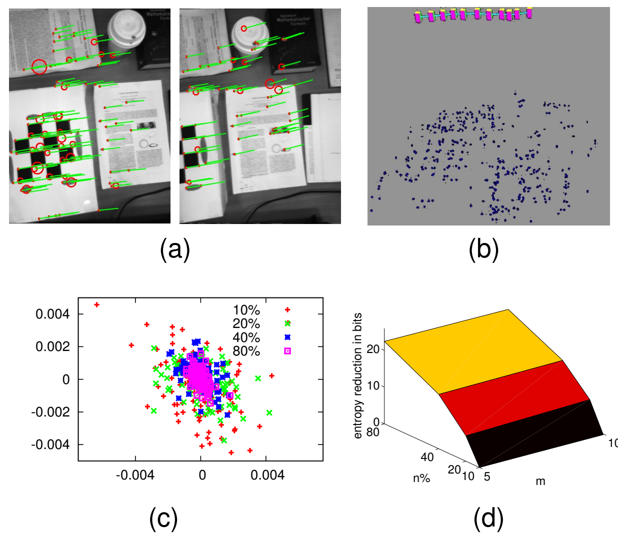


Fig. 5. Real image experiments. In (a), two keyframes of a sample image sequence are shown. SURF and FAST features which were matched to their corresponding 3D structure are highlighted. In (b), one can see the bundle adjusted motion and structure of the sample sequence. The distributions over the projections of the final camera positions $\mathbf{x}_m$ are shown in (c). Each distribution consists of BA estimation which either used 10%, 20%, 40%, or 80% of the available 3D points. One can see, that all four distribution share approximately the same mean. In (d), the influence of the percentage $n\%$ of the total number of features and the number of frames m on motion estimation accuracy is shown in terms of average entropy reduction over the five image sequences.

by the help of FAST [22] and SURF-guided patch matching. If the distance of the current camera pose to the nearest keyframe exceeds a threshold, a new keyframe is dropped and the whole system is bundle adjusted (Fig. 5 (b)).

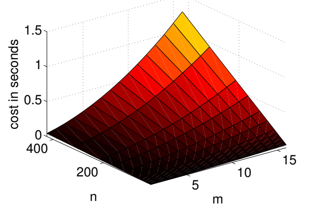
We performed a series of experiments similar to the sideward motion case. The camera was moved roughly sideways on a 25 cm trajectory 35 cm above an office desk. Although the scene was roughly planar, no such assumption was built into the system. An initial guess of the camera pose in the first frame was calculated from a known calibration target. As a reminder, we are concentrating on the building blocks of monocular SLAM in this evaluation, and not on the problem of structure and motion initialisation. In this spirit, the calibration grid mimics the previously estimated 3D structure of a longer SLAM sequence. Again, we evaluate the performance of the SLAM system by means of the error of the last camera pose in the trajectory. At this point, the calibration target has left the field of view. Hence, the pose accuracy purely reflects the performance of the SLAM framework used.

We performed the evaluation on five different image sequences. In each sequence, approximately 350 points were present. The full trajectory consists of $m = 10$ keyframes. However, each 3D point was only visible on average in three to four keyframes due to various reasons such as the restricted field of view, occlusion and the sensitivity of SURF and FAST to perspective transformations. For our analysis, we drew in each case 100 sample sets consisting of 80%, 40%, 20%, and 10% of the total number of 3D points and performed BA for each sample set. The resulting scatter in the final camera location estimate over the sample set is shown in Fig. 5 (c). One can see that the variance increases when the percentage of features goes down and that the distributions share approximately a common mean. In addition, we produced similar sample sets, where only $m = 5$ keyframes where considered. Due to the lack of ground truth, we consider the mean of the most accurate setting $\langle m = 10, n\% = 80\rangle$ as ground truth. Now the error of each single sample can be computed as well as the corresponding covariances over each sample set. As in the simulation experiments, each setting $\langle m, n\%\rangle$ is compared to the least accurate case $\langle 5, 10\%\rangle$ in terms of entropy reduction. The average entropy reduction in bits over the five image sequences is shown in Fig. 5 (d). One can clearly see that the shape of the surface is comparable to our simulation results.

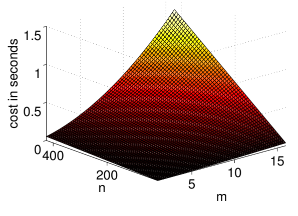
D. Computational Cost of Bundle Adjustment

In order to measure the computational cost of BA, we performed a Monte Carlo experiment similar as the ones above, but now measured the execution time of the high-performance SBA library for multiple experiments at each choice of parameters. We restricted each optimisation to 20 iterations in order to have comparable results. The resulting plot is shown in Fig. 6(a). Although in theory the computational cost of BA is cubic in the number m of keyframes, it is actually

$$O(m^2 \cdot n) \tag{11}$$

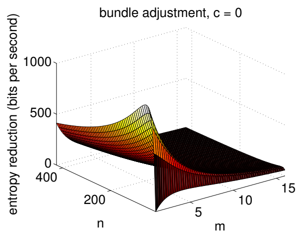


(a)

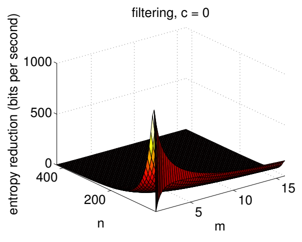


(b)

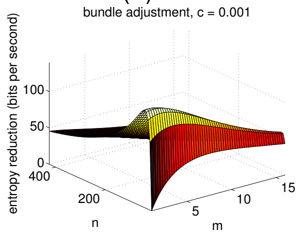
Fig. 6. Computational cost of BA. (a) shows the sparse data from Monte Carlo experiments, and (b) shows the fitted polynomial function.



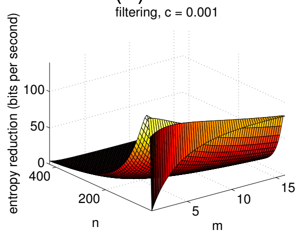
(a) BA



(b) EKF



(c) BA



(d) EKF

Fig. 9. Accuracy/cost measure in bits per second. In (a-b), no tracking cost is considered. In (c-d), a moderate tracking cost of $c = 1$ ms is taken into account

in our domain [9][13]. Efficient BA implementations typically consist of three mayor parts: calculation of the residuals and Jacobians (linear in the number of observations $m \cdot n$), building the linear system (quadratic in m and linear in n), and solving the linear system (cubic in m). Since in our domain the number of keyframes is relatively small, the cost of solving the linear system is negligible. Thus, calculating residuals/Jacobians as well as building the linear system dominates the computation and the cost is as stated in (11).

This leads to the polynomial function

$$\text{cost}_{\text{ba}}(n, m) = \beta_1(m + \beta_2)^2 \cdot n + \beta_3 \tag{12}$$

which represents the measured relation well. Parameter vector β was estimated using least-squares regression. With an NRMSE of 0.00144, the polynomial fits the data of the Monte Carlo experiment well (see Fig. 6(b)).

E. Computational Cost of Filtering

For EKF-SLAM it is well known that the computational cost per frame is $O(n^2)$ where n is the total number of landmarks in the map. This corresponds to the multiplication of the full covariance matrix with the Kalman innovation. However, in our domain where all landmarks in the state are visible at each step, the inversion of the mutual information matrix dominates. This leads to a computational demand of

$$O(n^3)\ . \tag{13}$$

This is equivalent to the cost of recovering the covariance from the information matrix (as in Eade and Drummond's system). Hence, we model the computational cost of filtering per frame using a polynomial of order three. In a Monte-Carlo experiment, the computational cost of the EKF update step was measured in Davison *et al.*'s publicly available real-time monocular SLAM implementation [7]. Again, the unknown parameters were fitted using non-linear least-squares regression (see Fig. 7(a)). The NRMSE was 0.00127. It is obvious that the computational cost for the whole trajectory is linear in the number of frames m.

F. Cost of Feature Tracking

SLAM estimation, whether filter updates or BA optimisation, is obviously a major cost in a real-time SLAM system. However, another significant factor is certainly the cost of feature tracking — which can include feature extraction, construction of descriptors, feature matching, active search, pose refinement from known structure, and/or global consensus matching methods such as RANSAC, active matching [3], or the joint compatibility test [19] to prune outliers. It is clear that the cost of matching over the whole trajectory is at least linear in the number of features. On the other hand, the matching cost is independent of the parameter m — the number of frames used for the SLAM optimisation. To clarify this, let us consider Klein and Murray's SLAM system. While only a small number of keyframes is used for the optimisation, the feature tracking runs on a much higher frame-rate. If one doubled the number of keyframes in the system, the computation time of the tracking thread would not change at all.

G. Trade-off between Accuracy and Cost

Finally, we need to weigh up accuracy against computational cost. By dividing (10) by (12) we get a combined accuracy/cost measure in terms of bits per second for the BA case. A similar measure is defined for the filter case. The results are shown in Fig. 9 (a-b). Even though the surfaces look different for the BA and filter, they share a common pattern. For all $n \in \{12, \dots, 425\}$, the accuracy/cost value is maximal at $m = 1$. This is a distinctive result: in order to increase the accuracy — independent of the choice between BA and filter — it is always more beneficial to increase the number of features than the number of frames in terms of computational cost. This pattern changes slightly if we include a moderate tracking cost of $c = 1$ms (per feature for the whole trajectory) into the analysis as can be seen in Fig. 9 (c-d). In the BA case, the ridge is shifted to the right.

To analyse this phenomenon in more detail, we examine contour lines of the accuracy surfaces. By solving (10) for

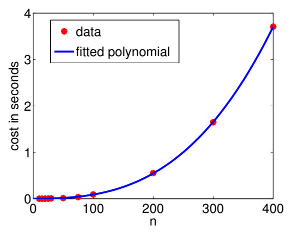


Fig. 7. The computational cost of EKF-SLAM per frame.

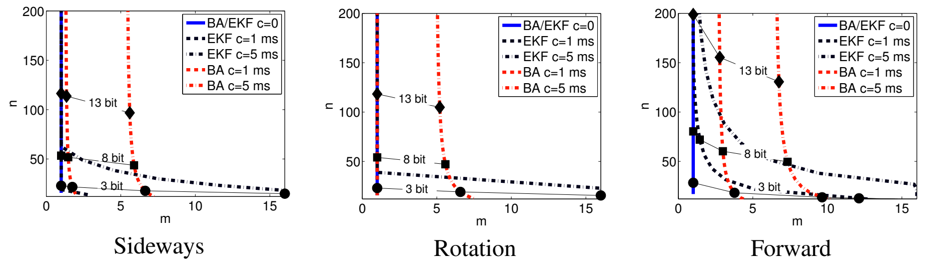


Fig. 8. Best choice of m and n to reach a specific accuracy at minimal cost. Three different accuracy levels (3 bit, 8 bit, and 13 bit) are highlighted.

m, we obtain the mapping $c_v(m) \rightarrow n$,

$$c_v(m) = \exp\left(\frac{v - \alpha_1 \log(m + \alpha_4) - \alpha_3}{\alpha_2}\right) - \alpha_5 \ , \quad (14)$$

which specifies the contour line for a specific accuracy level v. In order to find out which of these pairs $\langle m, c_v(m)\rangle$ (that all share the same accuracy v) is superior computationally, we apply the appropriate functions for computational cost and search for the minimum:

$$\langle m_v^*, n_v^*\rangle = \underset{\langle m,n\rangle \in \{\langle m, c_v(m)\rangle : m=1,\ldots,16\}}{\operatorname{argmin}} \operatorname{cost}(m,n) \ . \quad (15)$$

For a range of accuracy values v and different computational cost functions, the resulting optimal pairs $\langle m_v^*, n_v^*\rangle$ are plotted in Fig. 8. Even though the curves look different for different motion types, the overall pattern is the same. First of all, if we do not take tracking cost into account, the curves are vertical lines at $m = 1$. However, if a tracking cost is included, at low accuracy it is computationally beneficial to use more then one frame in the SLAM estimation. If we increase the accuracy more and more, each curve converges to a constant number of frames m whereas the number of features n increases monotonically. For filters, this constant m is one whereas for BA it is some value greater than or equal to one depending on the tracking cost. This pattern can be explained by the fact that for low accuracy values the tracking cost is relatively high compared to the total cost. Since the tracking cost is independent of the number of frames involved in the optimisation (as discussed in Sec. V-F), it is beneficial to use more then one frame in the optimisation even though the gain in accuracy is small. However, if the desired accuracy is high, the SLAM optimisation costs dominate, so that increasing the number of features is preferred over increasing the number of frames.

Finally, BA optimisation and filtering are compared directly in Fig. 10. Only at low accuracy (less then 3 bits improvement on the worst case) is the EKF computationally more efficient. This holds for the the case where we consider no tracking cost as well as when a tracking cost of $c = 1$ms is added. This seems to be the only region where filtering-based monocular SLAM systems remain competitive. For higher accuracy levels, BA is clearly more efficient. This is mainly because the cost of BA is linear in the number of features, whereas it is cubic for the EKF.

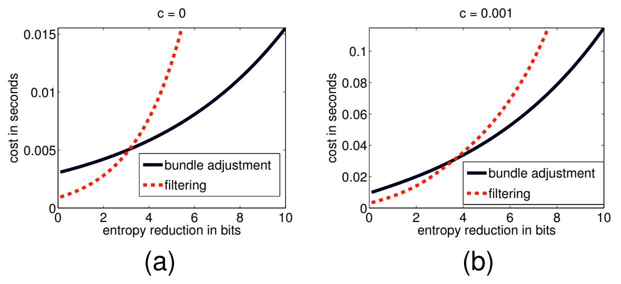


Fig. 10. Accuracy and cost for BA and filtering. No tracking cost is taking into account in (a), whereas a moderate tracking cost of $c = 1$ ms is considered in (b). The figures reflect the forward motion. Sideways and rotation leads to similar plots.

VI. Discussion

The simulations show that the only case when it might be worth filtering is when the tracking cost is high and the overall processing budget is small. One might reason that this is the operating point at which Davison's first real-time monocular SLAM system [6] was demonstrated seven years ago, although we have not made an analysis of absolute processing levels. Now with much more processing power available, it is not surprising, when looking at our accuracy/cost analysis, that the most accurate real-time monocular SLAM system available [14] is based on bundle adjusting a sparse number of keyframes.

To sum up, filters appear beneficial for small processing budgets whereas BA is superior elsewhere. However, before drawing too general conclusions from our analysis, we need to remember carefully which assumptions were made. First, we assumed that the accuracy of a filter-based SLAM estimate equals the accuracy of bundle adjustment. But, this only true in the best case. In particular, we did not include the problems of linearisation errors in our simulation. Our analysis also revealed that it is computationally most efficient to filter only a single keyframe (except when tracking cost is very high). Practically, this is not a good idea. It is well-known that linearisation errors are especially dominant if filters are employed at low a frame rate. These shortcomings in the presented experimental setup can only show filter-based monocular SLAM in an even worse light. Thus, an experiment which models the accuracy of filter-based SLAM more realisticly could only strengthen our conclusions.

On the other hand, we did not focus on all aspects of SLAM in our analysis. We intentionally did not consider large-scale SLAM and loop-closing since these issues have been intensively studied in the past. A SLAM frameworks which works reliably locally, whether it is BA or filtering, can easily be applied to a large scale problems using methods such as sub-mapping [21] or graph-based global optimisation [15]. Furthermore, it was shown recently that loop-closing can be solved efficiently using appearance-based methods (e.g. [5], [1]) which can be formulated independently from metric SLAM systems. Thus, we assume in our analysis that the choice between BA and filtering is not relevant at this global level.

In addition, we did not analyse the hard problem of bootstrapping camera tracking. Once a monocular SLAM system is initialised, the first guess $\mathbf{p}_0$ required by the Levenberg Marquardt algorithm in BA can be obtained easily using pose tracking with known structure [13]. However for jointly initialisation of structure and motion, this first guess $\mathbf{p}_0$ need to be inferred using a different method such as the $\{5, 7, 8\}$-point algorithm [20], [11]. We are not aware of any monocular SLAM framework which can solve the initialisation problem robustly under general configurations. The system of Klein and Murray [13] currently requires a separate initialisation step for the first two keyframes using a motion with parallax. With the right parameterisation (e.g. [4]), filtering can immediately give estimates along with uncertainty measures. Whether BA plus $\{5, 7, 8\}$-point algorithm or filters are better for the initialisation stage of monocular SLAM is an open question.

VII. Conclusion

To the best of our knowledge, we have presented in this paper the first detailed analysis of the relative merits of filtering and bundle adjustment for real-time monocular SLAM in terms of accuracy and computational cost. We performed a series of experiments using covariance propagation and Monte Carlo simulations for motion in a local scene. The experiments showed that in order to increase the accuracy of monocular SLAM it is more profitable to increase the number of features than the number of frames. We also performed real image experiments using a keyframe based SLAM framework and these are in accordance with the simulation results. Furthermore, we have introduced a combined accuracy/cost measure to permit absolute comparisons. While it is computationally most efficient to use a minimal number of frames and many features, this pattern changes slightly if tracking cost is included in the analysis. With the reservations discussed above, we can conclude that filter-based SLAM frameworks might be beneficial if a small processing budget is available, but that BA optimisation is superior elsewhere. These results are shown to be remarkably independent of the particular motion and scene setting.

Although this analysis delivers valuable insight into real-time monocular SLAM, there is space for further work. In particular, a comparison between filtering and BA during SLAM initialisation would be profitable and our instinct says that filtering may be more beneficial in this situation of high uncertainty.

Acknowledgements

We are very grateful to our colleagues at Imperial College London and the University of Zaragoza for discussions and software collaboration.

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] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[3] M. Chli and A. J. Davison. Active Matching for visual tracking. *Robotics and Autonomous Systems*, 57(12):1173 – 1187, 2009. Special Issue 'Inside Data Association'.

[4] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics (T-RO)*, 24(5):932–945, 2008.

[5] M. Cummins and P. Newman. Probabilistic appearance based navigation and loop closing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

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

[7] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007.

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

[9] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Proceedings of Photogrammetric Computer Vision*, 2006.

[10] C. G. Harris and J. M. Pike. 3D positional integration from image sequences. In *Proceedings of the Alvey Vision Conference*, pages 233–236, 1987.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[12] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics (T-RO)*, 24(6):1365–1378, 2008.

[13] 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.

[14] G. Klein and D. W. Murray. Improving the agility of keyframe-based SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

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

[16] M. I. A. Lourakis and A. A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009.

[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] 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.

[19] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[20] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[21] 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.

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