Double Window Optimisation for Constant Time Visual SLAM

Hauke Strasdat[a] Andrew J. Davison[a] J.M.M. Montiel[b] Kurt Konolige[c]

[a]Department of Computing, Imperial College London, UK
{strasdat,ajd}@doc.ic.ac.uk

[b]Instituto de Investigacion en Ingeniera de Aragon (I3A), Universidad de Zaragoza, Spain
josemari@unizar.es

[c]Willow Garage Inc, Menlo Park, USA
konolige@willowgarage.com

Abstract

We present a novel and general optimisation framework for visual SLAM, which scales for both local, highly accurate reconstruction and large-scale motion with long loop closures. We take a two-level approach that combines accurate pose-point constraints in the primary region of interest with a stabilising periphery of pose-pose soft constraints. Our algorithm automatically builds a suitable connected graph of keyposes and constraints, dynamically selects inner and outer window membership and optimises both simultaneously. We demonstrate in extensive simulation experiments that our method approaches the accuracy of off-line bundle adjustment while maintaining constant-time operation, even in the hard case of very loopy monocular camera motion. Furthermore, we present a set of real experiments for various types of visual sensor and motion, including large scale SLAM with both monocular and stereo cameras, loopy local browsing with either monocular or RGB-D cameras, and dense RGB-D object model building.

1. Introduction

Visual systems for mapping and localising in an environment are a key technology for many real-world applications, such as automatic driving service robotics, augmented reality and object modeling. There are two general purposes that map-making serves: accurate localization within a local area for tasks such as obstacle avoidance or object manipulation; and global connection information for longer-range navigational planning. One proposal for how maps could work in both cases is to consider them as *manifolds*; that is, locally Euclidean but globally topological [4].

Current work in visual mapping (VSLAM) has concentrated on either locally accurate reconstruction (e.g. PTAM [6]), or large-scale reconstruction with infrequent loop closures, typically in outdoor spaces and not in real time [8, 17]. In this paper, we present a system for real-time visual mapping that scales appropriately from small, accurate mapping of immediate space, to the rapid exploration of larger areas necessary for long-range planning. Our system borrows from the idea of a manifold, with two types of constraints. Within the local area of the current pose, an inner window of frames and points are bundle adjusted to give an accurate metric representation. This inner window is connected to a set of softer frame-frame constraints, the outer window, that constitute the larger graph of space.

Our approach is related to relative bundle adjustment (RBA) [16], which encodes constraints in local coordinate systems, and never attempts a global reconstruction. RBA has the desirable property that loop closures of both large and medium loops have effects that dissipate quickly in the graph, leading to constant-time updating. However, RBA does not enforce metric consistency *within* its optimisation window, and therefore it remains unclear how RBA could deal with loopy motion in a local area (as in Figure 3) without degrading in accuracy. In our system, bundle adjustment in the local area assures an accurate solution. Our algorithm automatically builds a suitable connected graph of keyframes and constraints, dynamically selects inner and outer window membership, and optimises both. Therefore, our approach shares similarities with the real-time stereo SLAM framework of Lim *et al.* [11] who alternate BA in a local window with global segment optimisation. In contrast to this, we minimise the error in both windows simultaneously using a common cost term, allow constant-time performance by restricting the size of the outer window, and also cover monocular SLAM. We demonstrate in extensive simulation experiments that our method approaches the accuracy of off-line bundle adjustment while maintaining constant-time operation, even in the hard case of very loopy monocular camera motion.

In the rest of the paper, we discuss further related work, and provide some insights into the difficulties of constant-time optimization using current techniques. We then present our scalable optimization technique using a double window approach, and show how it works for both stereo and monocular cameras. To validate the approach, we present real-world experiments for various types of visual sensors and motion, for both monocular and binocular devices.

2. Optimisation for Visual SLAM

Full bundle adjustment in VSLAM, while improving rapidly in absolute computational time [5, 7], still suffers from linear to cubic time in the number of variables (depending on particulars of the system), thus limiting its use in large-scale operation. For example, PTAM [6] runs full bundle adjustment in a background thread, which limits its scale to small workspaces. Our design goal is to have the same accuracy as PTAM in small workspaces while also scaling much better than full BA in handling rapid exploration. There are three main techniques that have been used in the past to tackle this issue:

- Active windows
- Relative representations
- Pose-pose reduction

Active Windows In order to achieve constant-time operation in a visual SLAM system, it is common practise to dynamically define a sub-set $M_{\mathcal{A}}$ of all M keyframes as the 'active window' $\mathcal{A}$ over which to apply optimisation. In visual odometry/sliding window BA frameworks designed for exploration, the active window will often consist of the $M_{\mathcal{A}}$ most recently captured frames. In general, keyframes are fixed at the boundary of the active window, and all those keyframes with co-visible points are typically included [13].

Fixing keyframes works well in exploratory situations with large but few loops (see Figure 1(a)). However, for a loopy camera motion (Figure 1(b)), the number of keyframes at the boundary is relatively large with respect to the total number of keyframes within the active window, and fixing them hampers convergence.

Relative Representations Relative Bundle Adjustment (RBA) [16] uses a relative representation for frame and point variables. Since the global position of the frames is not computed, it must be recovered from the relative variables, and involves significant computation. However, Sibley et al. [16] argue that there is usually no need for full reconstruction, and this argument aligns with our notion of a manifold, in which metric reconstruction occurs only in a local region. To work in constant time, RBA makes the active window assumption.

RBA is equivalent to BA if the network of relative constraints forms a tree. Thus, it works especially well on exploratory scenarios where there are no cycles within the active window (Figure 1(a)). However, the accuracy of RBA

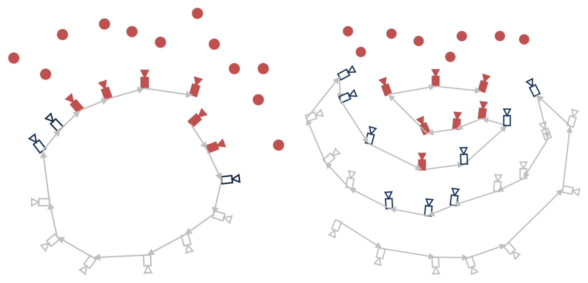


Figure 1. **Active windows.** Keyframes within the active window are red (filled); keyframes at the boundary are blue (dark, unfilled); inactive keyframes are grey (light, unfilled). In (a), the camera performs exploration around a loop. The active window has two open ends. In (b), there is very loopy browsing motion. Here, the boundary of the active window consists of many keyframes.

degrades if there are many loops, as it does not enforce the condition that constraints around the loop add up to identity.

Pose-graph Optimisation Instead of bundle adjusting all frames and points, the points can be marginalised out to yield constraints between the frames, for both stereo [8] and monocular systems [18]. The resultant system is an approximation, because binary links between frames do not fully encode the nonlinear connections between frames and points. Pose graphs do not reduce the computational complexity of the problem, since (again depending on the particulars of the graph) they have linear to cubic complexity.

3. Preliminaries

Before going on to present our new solution, we briefly define notation for the standard systems models we use throughout.

3.1. Camera poses

For stereo SLAM, we represent camera poses $\mathtt{T}$ as members of the Lie group SE(3), which consists of a 3×3 rotation matrix $\mathtt{R}$ and a translation 3-vector $\mathbf{t}$. In order to deal with scale drift, we represent the camera poses $\mathtt{T}$ for monocular SLAM using the Lie group of similarity transformations Sim(3) [18]:

$$\mathtt{T} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} \quad (1)$$

Here, the additional parameter s represents the local scale.

We use the following convention: An absolute pose $\mathtt{T}_i$ describes the transformation from the world frame into the camera frame i. A relative pose $\mathtt{T}_{ij}$ between two camera frames i and j is defined as $\mathtt{T}_{ij} := \mathtt{T}_i \cdot \mathtt{T}_j^{-1}$.

3.2. Monocular and Stereo Camera Models

For monocular SLAM, we use the standard pinhole camera model:

$$\hat{\mathbf{z}}_m(\mathtt{T}_i, \mathbf{x}_k) = \begin{pmatrix} f_1 \frac{y_1}{y_3} + p_1 \\ f_2 \frac{y_2}{y_3} + p_2 \end{pmatrix} \quad \text{with} \quad \mathbf{y} := \mathtt{T}_i \cdot \mathbf{x}_k, \quad (2)$$

where $\mathbf{f}$ is the focal length and $\mathbf{p}$ the principal point.

In binocular SLAM, each observation $\mathbf{z}_b$ is a 3-vector, where the first two components u_l, v_l are the pixel measurements in the left camera (reference frame). The third component u_r is the row measurement in the right camera frame. Thus, we get the following prediction function:

$$\hat{\mathbf{z}}_b(\mathtt{T}_i, \mathbf{x}) = \begin{pmatrix} \hat{\mathbf{z}}_m(\mathtt{T}_i, \mathbf{x}_k) \\ f_1 \frac{y_1 - b}{y_3} + p_1 \end{pmatrix} \quad \text{with} \quad \mathbf{y} := \mathtt{T}_i \cdot \mathbf{x}_k \ , \quad (3)$$

where b is the baseline. We assume all images are undistorted and rectified thanks to prior calibration.

4. Double Window Optimisation Framework

In this section, we introduce our scalable back-end for visual SLAM. For an example, we will concentrate on stereo SLAM. In Section 6, we explain how this framework is extended to monocular SLAM — including appropriate treatment of scale drift in constant time.

4.1. Overview

In order to achieve scalable, usually constant-time performance, we apply an active window scheme. The novelty of our framework is the fact that we use a *double window* approach. An inner window of point-pose constraints (i.e. bundle adjustment) is supported by an outer window of pose-pose constraints (i.e. pose graph optimisation). Pose-pose constraints are defined by *covisiblity* [12]. Two poses are connected to each other, if they share enough common features. Unlike some previous approaches which apply BA and pose-graph optimisation alternately [18, 11], we couple the point-pose constraints and the pose-pose constraints within a single optimisation framework. While the inner window serves to model the local area as accurately as possible, the pose-graph in the outer window acts to stabilise the periphery. The soft constraints of the periphery contrast with fixed keyframes within a (relative) BA approach, which are hard constraints.

4.2. The SLAM Graph Structure

The SLAM graph consists of a set of keyframe vertices $\mathcal{V}$, a set of 3D points $\mathcal{P}$, and a set of relative edges $\mathcal{E}$. Each keyframe vertex V_i saves its absolute pose $\mathtt{T}_i$, remembers which points $\mathbf{x}_k \in \mathcal{P}$ are visible from $\mathtt{T}_i$ and also saves all corresponding observations $\mathbf{z}_{ik}$. An edge E_{ij} between two pose vertices V_i and V_j has a covisibility weight w_{ij}, which

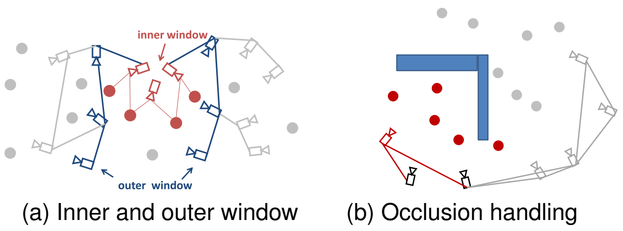


Figure 2. Illustrations of the Double Window Optimization (DWO) framework.

is the number of points which are visible both in V_i and V_j. Also, an edge is marked as being marginalised or not. If it is marginalised it also stores the relative pose constraint $\mathtt{T}_{ij}$ between $\mathtt{T}_i$ and $\mathtt{T}_j$. Otherwise, the relative pose is implicitly defined as $\mathtt{T}_{ij} = \mathtt{T}_i \cdot \mathtt{T}_j^{-1}$. A sample graph is visualized in Figure 2(a). At all times, there is exactly one *reference keyframe* V_{ref} .

4.3. Optimisation and Marginalisation

To construct the double-window structure, we start from the reference keyframe V_{ref}, and perform a uniform-cost search over the neighbours of V_{ref}, in such a way that the neighbour with the highest covisibility weight w_{ij} is selected first. This is in contrast to Lim *et al.* [11] who define the cost in the graph using the geodesic distance. The first M_1 keyframes are considered as being part of the inner window $\mathcal{W}_1$, whereas the following M_2 keyframes are members of the outer window $\mathcal{W}_2$ (typically $M_1 << M_2$). All points visible from the inner window are included in the optimisation. Thus, all frames in the inner window $\mathcal{W}_1$, and some frames in the outer window $\mathcal{W}_2$ are connected with point-pose constraints $\mathbf{z}_{ik}$ to the set of points as is usually done in BA. In addition, all frames in the outer window are connected to their local neighbours using pose-pose constraints $\mathtt{T}_{ji}$ as done in pose-graph optimisation. This results in the following cost function:

$$\chi^2 = \sum_{\mathbf{z}_{ik}} (\mathbf{z}_{ik} - \hat{\mathbf{z}}(\mathtt{T}_i, \mathbf{x}_k))^2 + \sum_{\mathtt{T}_{j,i}} \upsilon_{ji}^\top \Lambda_{\mathtt{T}_{ji}} \upsilon_{ji} \quad (4)$$

Here, $\upsilon_{ji} := \log_{\mathrm{SE}(3)}(\mathtt{T}_{ji} \cdot \mathtt{T}_i \cdot \mathtt{T}_j^{-1})$ is the relative pose error in the tangent space of SE(3) and $\Lambda_{\mathtt{T}_{ji}}$ is the precision matrix of the binary constraint $\mathtt{T}_{ji}$. Instead of estimating this uncertainty accurately using proper marginalisation [8], we suggest to approximate $\Lambda_{\mathtt{T}_{ji}}$ coarsely:

$$\Lambda_{\mathtt{T}_{ji}} = w_{ij} \begin{bmatrix} \lambda_{\text{trans}}^2 \mathtt{I}_{3\times3} & \mathtt{0}_{3\times3} \\ \mathtt{0}_{3\times3} & \lambda_{\text{rot}}^2 \mathtt{I}_{3\times3} \end{bmatrix} . \quad (5)$$

While the rotational component λ_{rot} is a constant, the translational λ_{trans} component should ideally be proportional to the parallax of $\mathtt{T}_{ji}$ — the translation $\mathbf{t}_{ji}$ normalised by the average scene depth. This efficient approximation of $\Lambda_{\mathtt{T}_{ji}}$ leads to very accurate results (see Section 7.1). Furthermore, we were not able to reproduce significantly better results using proper marginalisation instead. We believe the reason for this is twofold: On the one hand, turning BA into a binary pose graph is an approximation per se, because the marginalisation of a landmark visible in N frames should ideally lead to an hyper-edge with $(N-1)6$ DoF. On the other hand, the pose-pose network we use embodies a covisibility graph with typically many inter-connections (such as in Figure 3). We believe that the accuracy supported by this structure overwhelms the approximation due to use of diagonal precision matrices.

Double window optimisation is performed by minimising the sum of squared error χ^2 with respect to all poses $\mathtt{T}_i \in \mathcal{W}_1 \cup \mathcal{W}_2$ in the double window and all corresponding points $\mathbf{x}_k$. First and second order sparsity is taken into account, and the optimisation is performed using a efficient state-of-the-art sparse graph optimizer $\mathbf{g^2o}$ [9]. During optimisation, we do not define a fixed origin, since fixing a keyframe as the global origin can seriously degrade convergence if the selected keyframe is badly localized relative to its neighbours. Instead, we let the damping factor of Levenberg-Marquardt takes care of the gauge freedom [5].

Before joint optimisation, we make sure that all pose and point parameters are well initialised. A pose which was optimised previously is considered as a good initial guess. Otherwise the pose $\mathtt{T}_j$ might be substantially wrong and need to be realigned to its local neighbourhood. Starting from the reference pose $\mathtt{T}_{\text{ref}}$, we initialise

$$\mathtt{T}_j = \pi_{ja} \cdot \mathtt{T}_{\text{ref}} \tag{6}$$

along the path of relative pose constraints π_{ja}. In order to make sure that the points are triangulated well, we perform a few iterations of structure-only optimisation, which can be done very efficiently since the point locations are independent given the poses. Instead of anchoring the 3D points with respect to a keyframe [16, 11], we express them simply in a global coordinate frame, and thus can avoid a point management overhead.

4.4. Candidate Points Set for Tracking

For pose tracking, we seek to detect a set of 3D points in the current image. The camera pose is then estimated by minimising the reprojection error between the detected features and the point reprojections. In PTAM, all 3D points in the map are potential candidates for tracking. However, this does not scale very well with the number of points in the map. In most large-scale/visual odometry-based SLAM frameworks, points from the last m keyframes are considered.

In our framework, we select points which are visible from the local neighbourhood around the reference keyframe V_{ref}. The local neighbourhood $\mathcal{N}_1$ consists of all keyframes V_i connected to V_{ref} including itself:

$$\mathcal{N}_1 := \{V_i : E_{\text{ref},i} \in \mathcal{E}\} \cup \{V_{\text{ref}}\}. \tag{7}$$

All points visible from these frames are considered as potential candidates for tracking. As in PTAM [6], a point is only used for tracking if its reprojection lies within the current image boundaries, it is not too far or too close, and is not seen from a too different viewing angle compared to its initial observation.

Apart from scalability, selecting points using the local neighbourhood of frames has another advantage over PTAM's approach: it implicitly takes care of occlusion (see Figure 2(b)). Points which are occluded in the current frame are probably not visible from nearby frames either.

4.5. Adding New Keyframes

Occasionally, the current video frame is added as a new keyframe V_i to the graph. For all keyframes v_j in the graph which share at least θ (typically θ being 15 to 30) covisible points with the current frame, we include an edge E_{ji}, mark it as unmarginialized and assign a correpsonding covisibility weight w_{ji}. Finally, the new keyframe V_i is chosen to be the new reference keyframe $V_{\text{ref}} := V_i$.

5. Loop Closures

We distinguish between two types of loop closures. The first type is local loop closures which can still be detected metrically. The second type is large loop closures which are detected using appearance-based loop closure detection techniques.

5.1. Metric Loop Closure

Checking for metric loop closures is done by searching for 3D points in the reference keyframe V_{ref} which are not visible from its neighbourhood $\mathcal{N}_1$ (see Eq. 7). First, we determine a larger neighbourhood $\mathcal{N}_2$ around V_{ref} using uniform-cost search (as described in Section 4.3). Then, we construct a set $\mathcal{A}$ of potential loop closure points by selecting points which are visible in $\mathcal{N}_2$, but not in $\mathcal{N}_1$. If enough of those points are found in the current frame, we minimise their reprojection error wrt. a common pose $\mathtt{T}_{\text{loop}}$ using robust refinement, starting from $\mathtt{T}_{\text{ref}}$ as the initial guess. Afterwards, we prune all points from $\mathcal{A}$ whose reprojection exceeds a threshold (e.g. one pixel).

Now, for each keyframe $V_i \in \mathcal{N}_2 \backslash \mathcal{N}_1$ we check how many points in $\mathcal{A}$ are also visible in V_i. If there are θ or more co-visible points between V_i and V_{ref}, we have detected a metric loop closure, and we include a new edge $E_{\text{ref},i}$. This edge is marked as being marginalised, and the corresponding pose constraint $\mathtt{T}_{\text{ref},i}$ is set:

$$\mathtt{T}_{\text{ref},i} := \mathtt{T}_{\text{loop}} \cdot \mathtt{T}_i^{-1}. \tag{8}$$

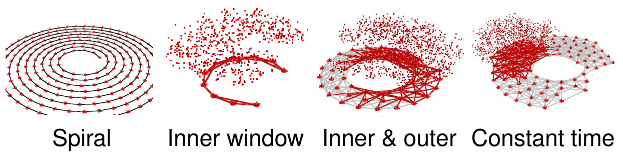


Figure 3. Spiral simulation scenario

Note that if $\mathtt{T}_{\text{loop}} \neq \mathtt{T}_{\text{ref}}$, the residual $v_{\text{ref},i} = \log(\mathtt{T}_{\text{ref},i}\mathtt{T}_{\text{ref}}\mathtt{T}_i^{-1})$ might be large, something which will be resolved in the next optimisation step (Section 4.3).

5.2. Large-scale Loop Closure

Candidates for large loop closures can be efficiently detected using appearance information only [2, 14]. These methods will detect a potential loop closure between the reference keyframe V_{ref} and and an older frame V_i. The loop closure can be efficiently verified using a 3-point RANSAC scheme. Given a set of three 3D-3D corresponences, a relative pose $\mathtt{T}_{i,\text{ref}}$ is uniquely defined [3]. If more than θ inliers are found, the loop closure is accepted and an edge $E_{\text{ref},i}$ is added the the graph. The matched 3D point pairs are merged into single points.

6. Extension to Monocular SLAM

In monocular SLAM there is a scale ambiguity. For BA, this has no particular consequences apart from the fact that the overall gauge freedom increases from 6 DoF to 7 DoF. However, in pose graph optimisation more care has to be taken. It can be shown that 6 DoF pose-pose constraints are not sufficient to correct for scale drift. Instead, 7 DoF pose constraints can be used to correct for rotational, translational and scale drift [18].

Since significant scale drift only occurs along large loops, and we are interested in a constant-time treatment of scale drift, we apply the following heuristic. The absolute poses $\mathtt{T}_i$ as well as the relative poses $\mathtt{T}_{ij}$ are members of Sim(3) instead of SE(3). However, the scale parameter s remains fixed most of the time. When a new keyframe is added to the graph, the corresponding scale is set to $s = 1$. There is only one case when a scale $s \neq 1$ is introduced: at large-scale, appearance-based loop closures (Section 5.2). Here, 3-point RANSAC recovers the rigid-body transformation SE(3) and the corresponding scale s [3]. This scale change gets propagated once poses are reinitialised using Eq. 6.

7. Experiments

7.1. Simulation Experiments

The main motivation for our double-window optimisation (DWO) approach is that it can deal with different motion patterns. In particular, it can smoothly handle both very

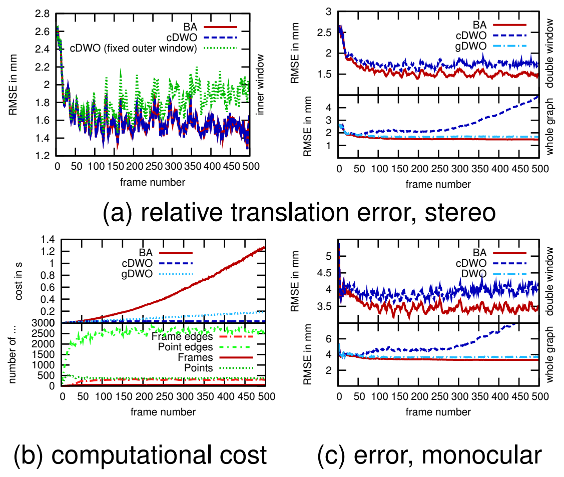


Figure 4. Spiral simulation experiment. The plot shows averages over ten Monte Carlo trials.

loopy local motion and large scale exploration. We evaluate a combination of both of these in our first set of Monte-Carlo simulation experiments. Here the camera moves in a spiral (see Figure 3), and the trajectory consists of 500 keyframes. We assume a stereo camera model with focal length of 300, a baseline of $5cm$, a resolution of 640×480, Gaussian image noise of one pixels and perfect data association. We compare BA over all frames to our double-window optimisation. Both methods are implemented using the efficient state-of-the-art sparse graph optimizer $\mathbf{g^2o}$ [9] and executed on a single core of an Intel(R) Core(TM) i7 960 desktop computer.

For each keyframe, we perform three iterations of joint structure and motion optimisation (BA or DWO). We apply two variants of the double window optimisation. The first variant (cDWO) is made to have strictly constant-time operation by restricting the inner window to 15 frames and the outer window to 50. In the second variant (gDWO) the outer window covers all remaining 485 frames and therefore allows global metric mapping.

In order to define an error measure, we have to remember that our scheme does not have a fixed origin, and therefore comparing absolute poses is meaningless. Instead we follow the approach of Kümmerle *et al.* [10] and define a relative error in terms of the relative differences $\Delta\mathtt{T}_{ij} := \mathtt{T}_i\mathtt{T}_j^{-1}$ between two absolute poses. In particular, we define the root mean square error (RMSE) over the difference of estimated and true relative translations,

$$\sqrt{\frac{1}{|\mathcal{E}|}\sum_{E_{ij}\in\mathcal{E}}\left(\Delta\mathbf{t}_{ij}^{[\text{est}]} - \Delta\mathbf{t}_{ij}^{[\text{true}]}\right)^2}, \tag{9}$$

with $\Delta\mathbf{t}_{ij}$ being the translational component of $\Delta\mathtt{T}_{ij}$. We analyse the RMSE at three different levels as shown in Fig-

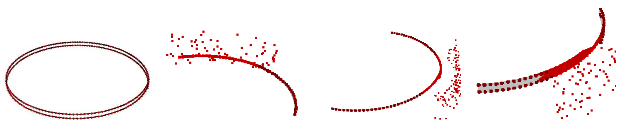
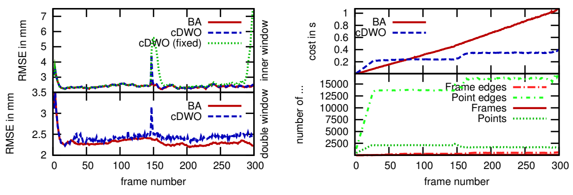


Figure 5. Double loop simulation scenario: At the start, the most recent 25 frames lie within the inner window, while the outer window is dragged behind. At loop closure, the inner window is at the center, while the outer window extends in both directions.



(a) Translation error (b) Computational cost

Figure 6. Double loop experiment. The plot shows averages over ten Monte Carlo trials

ure 4(a). First, we consider the local error within the inner window (left). One can see that the constant time framework (cDWO) reaches the same accuracy as BA. A second RMSE is computed at an intermediate level considering errors in both windows (top right). Once the 15th frame is passed cDWO slightly degrades from BA, but settles down quickly. Finally, we calculate a global error by considering all relative constraints (bottom right). Here, gDWO stabilises close to BA, while cDWO is clearly inferior since it only ensures accuracy within the double window.

Figure4(b) illustrates the computational cost for all three methods (BA, cDWO, gDWO). The constant computation times of cDWO can be well understood by studying the number of frames, points, point-to-frame constraints and frame-to-frame constraints used within the optimisation windows (bottom). We performed a comparable simulation experiment using a monocular camera. The RMSE is converted into scale-invariant version by normalising the translation vectors $\Delta\mathbf{t}_{ij}$ to length one. The corresponding accuracy plots are shown in Figure 4(c), forming a similar pattern than for stereo SLAM.

A second set of monocular simulation experiments is performed in order to demonstrate that our double window framework can deal with large loops and scale drift in a constant time fashion. The motion trajectory goes around a large loop twice as shown in Figure 5. At loop closure, an average scale drift of 1% is detected. The corresponding accuracy and cost is shown in Figure 6. Here, we have chosen an inner window size of 30 and an outer window size of 100. Note that the computational cost of cDWO slightly increases at loop closure (frame 150), simply because the number of visible point-to-frame constraints increases as

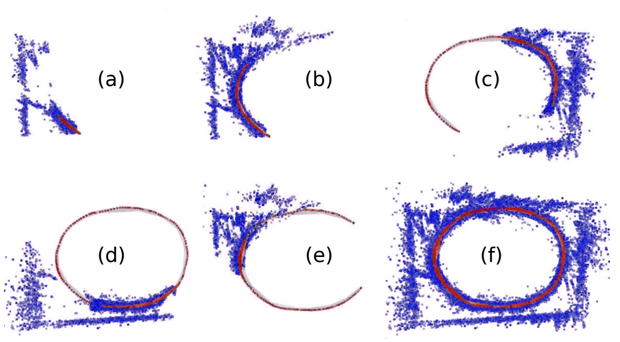
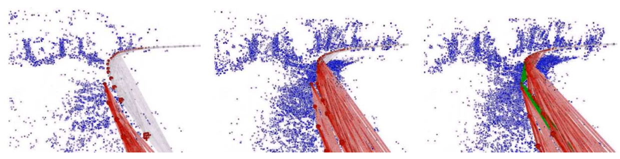


Figure 7. Large-scale loop closure, top view, stereo. See text.



(a) Right before (b) At (c) Right after
large-scale loop closure

Figure 8. Large scale loop closure in stereo SLAM: At loop closure, the active (red) region expands (b). Right afterwards, metric loop closures (green) are detected (c).

can be seen in Figure 6(b).

Finally, we adapted cDWO such that all frames in the outer window are fixed during optimisation. One can clearly see in Figure 4(a) left and Figure 6(a) top that the usage of such hard constraints lead to inferior results.

7.2. Real-image experiments

To further evaluate the DWO back-end, we have employed a range of real-image simulation experiments using stereo cameras, monocular cameras and an RGB-D camera. The experiments where performed on desktop computer with an Intel(R) Core(TM) 2 Duo CPU with 2.66 GHz

7.2.1 Stereo SLAM

We developed a stereo front-end including the following steps: First, a disparity image is calculated between the left/reference frame and the right frame using dense stereo matching. Second, we perform guided matching on 3D candidate points (which are constructed as described in Section 4.4). Afterwards, we use guided BRIEF [1] matching between measurements and candidate points seeded at FAST [15] corners – followed by 3D to 3D robust pose refinement to produce an initial pose estimate. Finally, we perform guided patch matching on image pyramid followed by a second robust pose refinement step.

We integrated the stereo front-end with DWO and preformed a large scale SLAM experiment using the New Col-

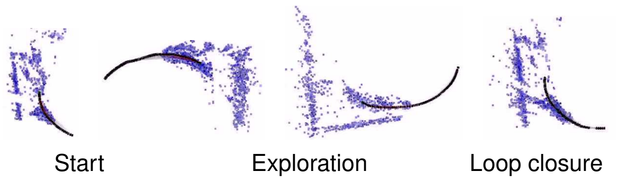


Figure 9. Monocular large-scale SLAM in constant time (17 FPS)

lege data set[1]. The estimated trajectory is illustrated in Figure 7. We have chosen an outer window size large enough to cover the whole loop. Figure 7(d) and 8(a) illustrate the situation right before loop closure. At loop closure (Figure 8(b)), both ends of the graph get connected, and the visible set of 3D points instantly amplifies. Afterwards (Figure 8(c)), both rounds of the loop are become interconnected using metric loop closure constraints. Also, note that after the loop closure, the total number of frames exceeds the size of the double window, so that the outer window forms a stabilising periphery around the inner window (Figure 7(e)). The computational performance of the stereo system is currently near real-time (5-7 FPS). Finally, by increasing the inner window size to the total number of frames, full BA can be performed (Figure 7(f)). This computation takes on the order of seconds.

7.2.2 Monocular Camera

Our monocular front-end is largely based on the PTAM tracker but adapted to the needs of the DWO back-end. 3D candidate points for tracking are estimated as described in Section 4.4. Epipolar search for feature initialisation is only performed between keyframes which are connected with an edge. In general, all `for` loops in PTAM which iterate over all points or all frames are replaced. Instead, sets of points and frames are accessed along the local connectivity of the SLAM graph. For the 3-point RANSAC step in appearance-based loop closure detection, SURF features are employed. Their depths are estimated using k-nearest neighbour regression [18]. Thus, the depth of a SURF point is calculated from the average depth of map points with nearby reprojections.

In the first experiment we demonstrate DWO in combination with the modified PTAM tracker for large scale monocular SLAM (Figure 9). We have chosen a strictly constant time setting, such that the inner plus outer window only covers approximately a quarter of the loop. At loop closure (Figure 9(d)), a scale change of 6% is detected and both ends of the graph are attached appropriately. The whole system runs at 17 FPS — applying the tracker and the optimisation alternately in a single thread. In a second monocular experiment, we demonstrate that our back-end can deal with loopy browsing motion (which is the speciality of PTAM), but also with rapid exploration (Figure 10).

[1] http://www.robots.ox.ac.uk/NewCollegeData/

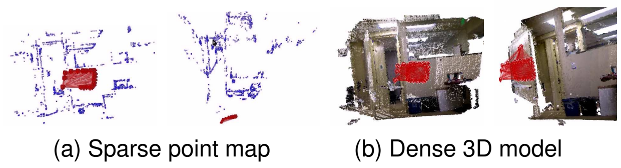


Figure 11. Loopy browsing motion in office environment, RGB-D

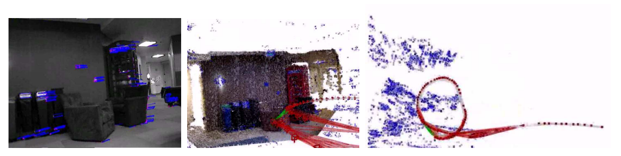


Figure 12. SLAM using a RGB-D camera on a wheeled robot.

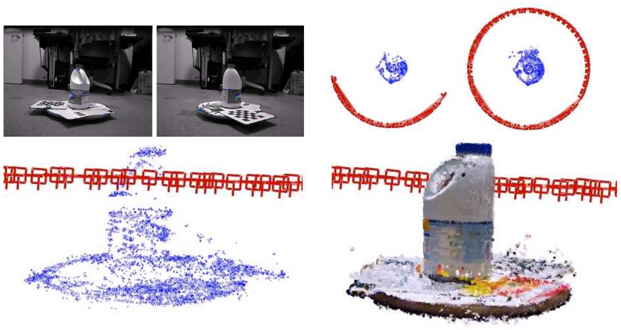


Figure 13. Dense object models using RGB-D camera. Bottom right: Sparse points clouds are used for SLAM. Bottom left: Raw RGB-D measurements are overlaid to create a dense object model.

7.3. RGB-D cameras

Our visual SLAM framework can be also used with RGB-D cameras that have become popular very recently. We used a PrimeSensor from PrimeSense which calculates a dense 3D cloud using structured light and is largely identical to Microsoft's Kinect. The PrimeSensor outputs an RGB image together with a registered 3D point cloud. As a first step, we transform the 3D point cloud into a disparity image registered to the RGB image. Thus we can use the RGB-D camera in the way as any other stereo camera and therefore, we use the very same front-end as summarised above.

Figure 11 illustrates loopy browsing motion using the RGB-D camera. The tracking and optimisation solely depend on sparse feature matching (a). However, the dense point clouds can be registered to the optimised frames and used to construct a dense environment model cheaply (b). Also, we attached the RGB-D camera to a wheeled robot and mapped of an indoor environment (Figure 12),

Finally, we demonstrate how our framework can be used to create dense object models (see Figure 13). An object

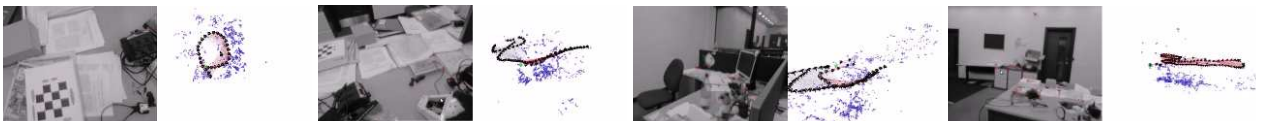


Figure 10. Loopy browsing motion plus exploration in office environment, monocular

is placed on a turning table and observed by a static RGB-D camera. We need to create an image mask which only covers the dense object. First, we remove the background by rejecting all pixels with a depth greater than a particular threshold. Secondly, we detect the ground plane and only accept pixels whose corresponding 3D points are significantly above it. Thirdly, the object mask is dilated in order to make sure that pixels at the object boundary are rejected. Note that the loop is closed using metric loop closures.

A number of videos are available online which illustrates our simulation and real-image experiments[2].

8. Conclusion

We have presented a novel double window framework for visual SLAM, which is unique in being able smoothly to cope with both detailed, loopy browsing, and rapid large-scale exploration in constant time, attaining the comparable results to full local bundle adjustment locally. Furthermore, we provided evidence that the use of soft constraints within a double window framework is superior to an active window approach with fixed frames. We have demonstrated our method with a wide set of informative simulation and real-image experiments, and hope that it can be adopted in many general purpose SLAM systems in the near future.

9. Acknowledgements

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

References

[1] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010. 6

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

[3] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987. 5

[4] A. Howard. Multi-robot mapping using manifold representations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1996. 1

[5] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1474–1481, 2010. 2, 4

[6] 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. 1, 2, 4

[7] K. Konolige. Sparse sparse bundle adjustment. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2010. 2

[8] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics (T-RO)*, 24:1066–1077, 2008. 1, 2, 3

[9] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 4, 5

[10] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009. 5

[11] J. Lim, M. Pollefeys, and J.-M. Frahm. Online environment mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 1, 3, 4

[12] C. Mei, G. Sibley, and P. Newman. Closing loops without places. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3738–3744, 2010. 3

[13] 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. 2

[14] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Accepted for oral presentation at CVPR 2006*, 2006. 5

[15] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006. 6

[16] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems (RSS)*, 2009. 1, 2, 4

[17] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *ACM Transactions on Graphics (SIGGRAPH)*, 2006. 1

[18] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale drift-aware large scale monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2010. 2, 3, 5, 7

[2] http://www.doc.ic.ac.uk/~strasdat/iccv2011/