Online Environment Mapping using Metric-topological Maps

Jongwoo Lim *
Google inc.
Mountain View, CA, USA
jwlim@google.com

Jan-Michael Frahm
Department of Computer Science
University of North Carolina
Chapel Hill, NC, USA
jmf@cs.unc.edu

Marc Pollefeys
Department of Computer Science
ETH Zurich, Switzerland
marc.pollefeys@inf.ethz.ch

October 30, 2011

Abstract

The paper proposes a vision based online mapping of large-scale environments. Our approach uses a hybrid representation of a fully metric Euclidean environment map and a topological map. This novel hybrid representation facilitates our scalable online hierarchical bundle adjustment approach. The proposed method achieves scalability by solving the local registration through embedding neighboring keyframes and landmarks into a Euclidean space. The global adjustment is performed on a segmentation of the keyframes and posed as the iterative optimization of the arrangement of keyframes in each segment and the arrangement of rigidly moving segments. The iterative global adjustment is performed concurrently with the local registration of the keyframes in a local map. Thus the map is always locally metric around the current location, and likely to be globally consistent. Loop closures are handled very efficiently benefiting from the topological nature of the map and overcoming the loss of the metric map properties as previous approaches. The effectiveness of the proposed method is demonstrated in real-time on various challenging video sequences.

1 Introduction

The progress of robotics and computing hardware in the last decade has increased the demand for online metric map reconstruction from cameras. At the same time the scale of those maps has been increased by two to three orders of magnitude [Agarwal et al., 2009, Frahm et al., 2010, Clipp et al., 2010]. This poses a significant challenge for current state of the art camera based large scale modeling approaches. One of the most demanding applications of vision based map reconstruction is in robotics. Robots inherently need to model the surround environment to safely navigate in the space while performing the various tasks. Our proposed approach for online simultaneous localization and mapping (SLAM) is tackling this problem which has also been intensively studied in the robotics community for decades.

Traditionally laser range finders (LIDAR) have been used in this task mainly because they directly measure the distance to the surface with high precision. However typical LIDAR sensors only scan a 2D slice of the space and the slice needs to be in the same plane for a SLAM system to work. This limits the use of laser-based SLAM systems in an environment with objects with complex height profile (such as tables or shelves) and for the robots such as MAV(micro aerial vehicle) or humanoids which move freely in 3D space, not restricted on a flat floor. Also these sensors are larger in size, heavier, and consume more power compared to passive sensors like video cameras.

Recently there have been many attempts to address SLAM for larger environment using monocular or stereo cameras [Davison et al., 2007, Konolige and Agrawal, 2008, Clipp et al., 2010]. By using cameras, the visual SLAM system can avoid the shortcomings of laser-based SLAM systems, but instead it needs to solve the

*Major portion of the work in this paper was done while the first author was with Honda Research Institute USA, inc. at Mountain View, CA, USA.

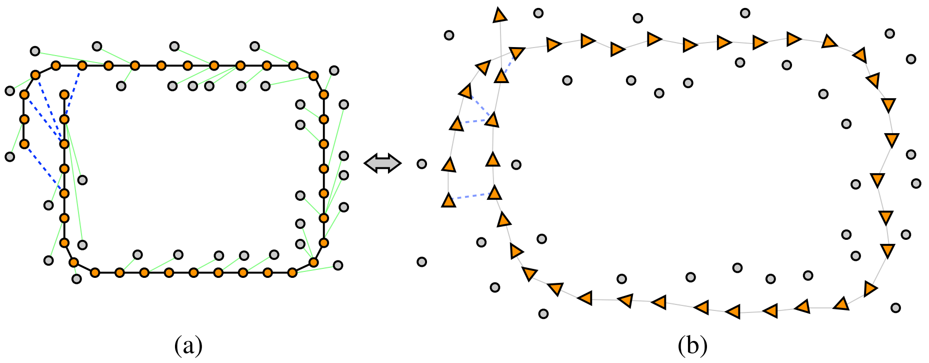


Figure 1: (a) A keyframe pose graph which shows the topological structure of the environment. Orange circles represent the keyframes and gray circles the landmarks. (b) A metric embedding of the keyframes together with the landmarks. Refer the text for detailed description.

problem of estimating the 3D environment from image observations. The recent advances in structure-and-motion research provide the theoretical basis and useful machineries [Pollefeys et al., 2004, Nistér et al., 2006].

In SLAM systems, the most difficult problem is to maintain the map (the perceived model of the environment) consistent to all observations, especially when loops exist in the robot trajectory. Traditionally in visual SLAM this map adjustment is performed by bundle adjustment which scales cubically with the problem size prohibiting online computation in large scale environments. Topological mapping is devised to avoid this problem. It represents the world as a graph with a set of places (nodes) and the relative location information between the places (edges). In this representation, a loop closure is one simple edge addition to the graph, and it does not require any additional adjustment. However, in return, it loses the globally metric property. For example, a robot can not perform spatial reasoning for proximity unless the link between the map locations is present in the topological map.

The proposed approach deploys a hybrid mapping method combining the benefits of metric Euclidean maps and topological maps, namely the locally-metric globally-topological mapping. The map is represented as a graph of the keyframes (nodes) and the relative pose between keyframes (edges), like the topological approach. The main distinction to previous approaches is that the system enforces the metric properties locally and globally as much as possible. The metric correctness in the map is the property that if one follows the links in a loop and combines their relative poses in order, the combined pose becomes the identity transformation, i.e. it exactly returns to the starting pose. Our method strictly enforces the locally metric property all the time via local adjustment, and globally metric property is achieved (with some delay) via a decoupled parallel 'global adjustment' process.

The paper is organized as follows; in the next section we review the related work. Our proposed novel method is described in detail in Section 3. More information on the design choices of the method are provided in Section 4 and Section 5 presents the experimental results on various challenging sequences. Finally we summarize the proposed method in Section 6.

2 Related Work

Triggs et al. [Triggs et al., 2000] brought bundle adjustment into the focus of the computer vision community about a decade ago. Since then bundle adjustment has become a topic of high interest in computer vision given the ability to perform large scale structure from motion from video [Pollefeys et al., 2004] or from Internet photo collections [Snavely et al., 2006]. Since the structure from motion process is inherently incremental, the result structure has to be adjusted to overcome drift [Chris Engels, 2006].

Essentially bundle adjustment poses the structure and motion estimation as an optimization problem, which parameterizes each camera with six degrees of freedom (DOF) for the translation and rotation (optionally with parameters for the camera intrinsics and radial distortion), and the 3D scene points with their three position parameters, in a fixed global coordinate system. The projection equations are then used to derive a set of non-linear residual equations which are then linearized through Taylor series expansion. This delivers a large and very sparse linear equation system that can be efficiently solved through a sparse Levenberg-Marquardt solver [Hartley and Zisserman, 2004]. Large scale reconstructions are challenging since the complexity of the bundle adjustment is at least cubic in the number of cameras plus a linear complexity in the number of points.

The challenge in scalability has been recently addressed by several researchers through a variety of methods. A very prominent approach is nested dissection [A.George, 1973, Lipton and Tarjan, 1979], which is a divide-and-conquer method to solve a set of sparse constraints as used in structure from motion. Nested dissection regroups the reconstruction problem into sub-maps, which conceptually can be solved independently except for the cross sub-map interactions through parameters that influence multiple sub-maps. Then the parameters of the global problem are sorted in two groups: one group of parameters that only influence one sub-map (intra-sub-map parameters) and the other parameter group effecting more than one sub-map (inter-sub-map parameters). This enables parallelism during the sparse Cholesky factorization in the inner loop of the sparse Levenberg-Marquardt solver, which provides a sizable speed-up. The concept was initially deployed in the field of photogrammetry to improve the efficiency of bundle adjustment [Brown, 1976].

A concept similar in spirit to nested dissection has been introduced by Ni et al. [Ni et al., 2007]. They proposed a hierarchical bundle adjustment to overcome the computational complexity by partitioning the bundle adjustment into connected sub-maps by modeling the parameters of each sub-map in a local coordinate system and also dividing them into intra-sub-map parameters and boundary variables. In contrast to nested dissection the sub-problems are solved till convergence and only then propagate the residual energy into the global solution achieving significantly higher convergence. In contrast to both of the above systems our proposed approach is modeling the inter-sub-map relationships through topological transformations (relative motions) effectively summarizing the constraints into the transformation. This provides a more compact representation of the boundary variables leading to a computationally more efficient solution of the global problem. Ni and Dellaert recently extended their approach [Ni and Dellaert, 2010] to a multi-level sub-map tree which represents the constraints between the different sub-maps through the graph edges.

Alternatively, Snavely et al. [Snavely et al., 2008] aimed at reducing the redundancy in the data by determining a set of essential cameras ("skeletal cameras") and removing all remaining cameras from the optimization. Our system reduces the number of cameras by selecting key-frames along the lines of the approach of Clipp et al. [Clipp et al., 2010], and in the global problem further reduction using segments of keyframes is employed.

In our method we encode the constraints between sub-maps through the topological transformations between the sub-maps. This is similar to the reduction of the parameter space introduced by Konolige and Agrawal in their FrameSLAM method [Konolige and Agrawal, 2008]. The reduction in FrameSLAM is achieved through a marginalization of feature constraints into 6DOF constraints between cameras with their corresponding uncertainties. In contrast to FrameSLAM our method does not depend on the linearization of the projection function to create the marginalized constraints, hence we do not suffer under the inaccuracy in the linearization.

Holmes et al. [Holmes et al., 2009] linearize the 3D features in the local coordinate system of the cameras to obtain a fast solution of the local bundle adjustment losing the metric property of the solution. In contrast our method achieves a metrically correct solution while maintaining the efficiency during the solution of the sub-maps.

Similarly the HMT-SLAM of Blanco et al. [Blanco et al., 2008] aims at large scale modeling in real-time. Blanco et al. propose to update a topological graph of metric sub-maps to a globally consistent map by optimizing the constraints of adjacent sub-maps (sub-maps which have a topological transformation defined in between them) through iterative optimization of the non-linear constraints. The latter process is similar to bundle adjustment but does not encode the information of the local metric sub-map as in our proposed map.

After briefly reviewing the previous work we now will introduce our method in more detail.

3 Online Mapping

3.1 Keyframe Pose Graph

The environment map is represented as a *keyframe pose graph*, whose nodes are the keyframes and edges represent the relative pose between two keyframes. More precisely, an edge $a \to b : P_{ab}$ represents a link from node a to node b with the associated 3D Euclidean transformation P_{ab} (P_{ab} is a 4×4 camera projection matrix with six 3D pose parameters, which is inverse of the camera motion matrix). As the pose graph is an undirected graph; if $a \to b : P_{ab}$ is in the graph, $b \to a : P_{ba}$ (where $P_{ba} = P_{ab}^{-1}$) is also in the graph. An example keyframe pose graph is shown in Fig. 1 (a). Please note that there intentionally is no global coordinate system in this representation, so the keyframe poses are only represented relative to other keyframes. The landmarks are attached to their *anchor keyframes* in which they are first observed (green lines in Fig 1 (a)). Each landmark's position is represented as a homogeneous 4-vector $\boldsymbol{x}$ in the anchor keyframe's coordinate system.

The map is incrementally constructed as the camera moves. Most keyframes are linked to the previous keyframes via commonly observed landmarks (these temporal links are shown as black lines in Fig. 1). When the robot visits previously mapped places, the location recognition module [Clipp et al., 2010] finds additional links between the current keyframe and the keyframe in the map, which creates loops in the pose graph (dashed lines in Fig. 1).

We build a metric embedding of the keyframe pose graph centered at a given reference keyframe a_0 as follows.

```
1. put the keyframe $a_0$ at the origin ($\hat{P}_{a_0} = I_{4\times4}$), and
   push a tuple $(0, a_0)$ into a priority queue $pq$.
2. repeat until $pq$ is empty.
   - pop $(d, a)$ with smallest $d$ from $pq$.
   - for each neighbor keyframe $b$ of $a$, $a \to b : P_{ab}$,
     - if $b$ is not in the embedding,
       - add the keyframe $b$ with the pose $\hat{P}_b = P_{ab}\hat{P}_a$.
       - put $(d + |P_{ab}|_G, b)$ into $pq$.
3. for each landmark $l$ and its anchor keyframe $c_l$,
   - add the landmark $l$ at the location $\hat{\boldsymbol{x}}_l = \hat{P}_{c_l}^{-1}\boldsymbol{x}_l$.
```

$\hat{P}_a$ denotes the pose of a keyframe a in the embedded space. $|P|_G$ denotes the norm of the translation component in P, thus d in (d, a) is in fact the geodesic distance from a_0 to a on the graph.

Conceptually this procedure performs weighted breadth-first search of the pose graph from the reference keyframe and embeds the keyframes into the Euclidean space according to the order. The landmarks are then embedded using their anchor keyframe's embedded pose. Fig. 1 (b) shows an example of the embedded keyframe pose graph. Note, that the embedded maps may look very different depending on the choice of the reference keyframes, and often a loop in the map does not remain as a valid loop in the embedded map. If there is metric inconsistency in the loop (when the combined transformation along the loop $\neq I_{4\times4}$), the accumulated error will break the farthest link from the reference keyframe.

3.2 Local Adjustment

A new keyframe adds new observations to the landmarks that are commonly visible from its nearby keyframes, and this may introduce large change in the map. The estimated pose of the new keyframe may not be very accurate due to various reasons, such as feature tracking error, outlier features, uncertainty in landmarks' 3D locations, etc. In our method the changes and inaccuracies in the map by the new observations are mostly handled by the local adjustment process. It improves the estimated pose of the current and nearby keyframes by minimizing reprojection errors in the local area, and at the same time ensures a locally metric property around the current location. In the case of a loop detection, the local and global metric consistency may also be violated by the newly added link. It is very important to resolve these inconsistencies quickly and at least locally, to ensure that the ongoing pose estimation and location recognition would work properly.

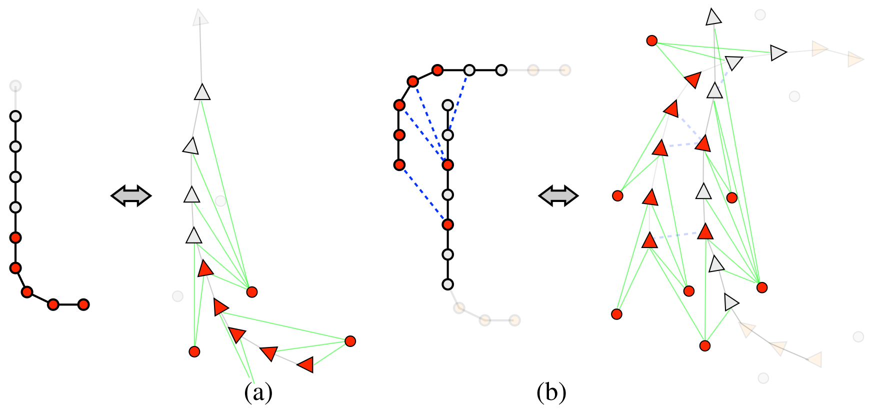


Figure 2: (a) Local adjustment. Active keyframes and landmarks are drawn in red, and fixed keyframes in light gray (window size=5). Green links show the observations of landmarks by keyframes. The fixed keyframes are those outside the window but with observations to the active landmarks. Dimmed keyframes and landmarks are not used in the local optimization. (b) Local bundle adjustment with a detected loop.

Our proposed local adjustment process updates the relative poses of the *active* keyframes around the current location and the positions of the active landmarks. The most recent-w keyframes (w is the window size parameter, typically 5$\sim$10) are initially selected as the active keyframes. We also include the keyframes connected by the loop-closure links to the recent-w keyframes into the active keyframe set. Next, all landmarks visible from the active keyframes are set as the active landmarks so that their positions will be updated in the optimization. All other keyframes which have observations of the active landmarks are included as *fixed* keyframes. Their (relative) poses are not changed by the local adjustment, and they act as anchors in updating the active keyframe poses and landmark positions. Note that the computational cost is mostly determined by the number of active keyframes (cubically) and active landmarks (linearly in SBA). In our method, the size of the active keyframe set is bounded by $2w$ since the location recognition only adds no more than one additional link per keyframe. Figure 2 provides visual illustration of the local adjustment.

Local adjustment performs the following steps for every new keyframe.

1. From the current keyframe, find the active keyframes $\{a_j\}$, then active landmarks $\{l_i\}$ and fixed keyframes $\{a'_k\}$.
2. Embed $\{a_j\}$ $\{l_i\}$ $\{a'_k\}$ into a metric space centered at the reference (most recent) keyframe $a_0 \in \{a_j\}$, having the keyframe poses $\{\hat{P}_{a_j}\}$ $\{\hat{P}_{a'_k}\}$ and landmark positions $\{\hat{\boldsymbol{x}}_{l_j}\}$ in the space.
3. Compute the improved active keyframe poses $\{\tilde{P}_{a_j}\}$ and landmark positions $\{\tilde{\boldsymbol{x}}_{l_j}\}$ using SBA.
4. Update the map from adjusted keyframe poses $\{\tilde{P}_{a_j}\}$ and landmark positions $\{\tilde{\boldsymbol{x}}_{l_j}\}$:
   - any existing $a \to b$, $a \in \{a_j\}$, set $a \to b : \tilde{P}_b \tilde{P}_a'^{-1}$.
   - any existing $b \to a$, $a \in \{a_j\}$, set $b \to a : \tilde{P}_a \tilde{P}_b'^{-1}$.
   - any $l \in \{l_j\}$ whose anchor keyframe is c_l, $c_l \in \{a_j\} \cup \{a'_k\}$, set $\boldsymbol{x}_l = \tilde{P}_{c_l} \tilde{\boldsymbol{x}}_l$.

Once the keyframes and landmarks are embedded into a metric space, the standard sparse bundle adjustment algorithm is used for optimization of the embedded parameters. In the embedded space the explicit topological structure between keyframes is not used anymore, but as in the standard SBA, the keyframes are connected via their landmark observations. After applying the Schur complement in SBA, the largest linear system to be solved has at most $12 \times w$ variables, which can be solved very quickly when w is small. The number of landmarks and fixed keyframes affects the performance through the increased number of observations and landmark update steps after pose estimation, but in a usual setup the local adjustment runs very efficiently. Finally the optimized keyframes and landmarks are converted back to relative representation in the environment map (the keyframe pose graph). Note that all anchor keyframes for the active landmarks are always part of the embedding as either active or fixed keyframes from the construction.

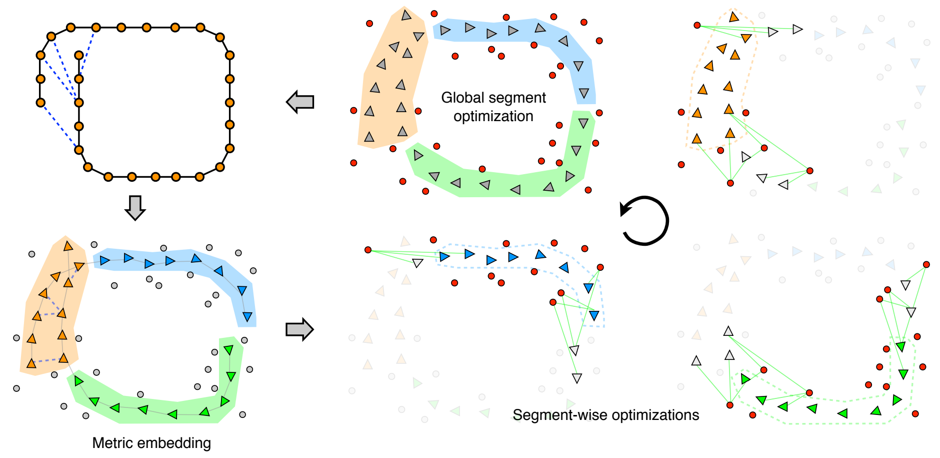


Figure 3: Global adjustment procedure. The keyframe pose graph is partitioned into multiple segments and the segments are embedded into a metric space. Each segment is optimized by the local adjustment algorithm if necessary, then the global segment optimization adjusts the segments' poses and landmarks' position assuming that the segments are moving rigidly. After iteration the result keyframe poses and landmark positions are updated back into the keyframe pose graph. The green lines in the segment-wise optimization illustration are shown to visualize how the fixed keyframes are selected.

While this process on first sight seems similar to RBA [Sibley et al., 2009] it shows advantages over RBA through the details. The most important improvement is that our method guarantees the local metric consistency since all entities are embedded in a metric space, whereas that is not guaranteed in RBA. Note that there is no simple way to enforce the metric constraint in purely topological framework like RBA. Moreover, in our approach there is no need to propagate Jacobian matrices over the edges of the graph, making it more computationally efficient. Additionally, our technique is conceptually simpler and also easier to implement than RBA since it fully operates in the familiar metric space and uses well established bundle adjustment methods.

3.3 Global Adjustment

With the above described local adjustment, the environment map is guaranteed to be locally metric, but still it may not be globally metric due to the errors along the loops[1]. In general achieving global metric consistency on a large map is not simple. Similar to the local adjustment, the approach we take is to embed the entire map into a metric space, optimize the embedded structure, and update the result back into the topological map. However we need to do this in more efficient and robust way. When a large number of keyframes and landmarks exist, using conventional SBA may take significant computation time and may have difficulty in converging to the correct geometry when there is large discrepancy along a loop.

In this section we propose a novel divide-and-conquer strategy to efficiently solve the global adjustment problem. First the keyframes are clustered into multiple disjoint sets (called *segments*) using geodesic distance on the graph, then the global adjustment module iterates local intra-segment optimizations and a global inter-segment optimization:

1. group the keyframes into κ segments $s_k = \{a_j^{(k)}\}$.
2. for each segment s_k, run intra-segment adjustment if necessary.
   - set all keyframes in s_k as active keyframes, and find active landmarks and fixed keyframes (in other segments).
   - embed all keyframes and landmarks, and run the local adjustment process.
   - update the active keyframes and landmarks.

[1]Note that if there is no loop in the graph, there will be no violation of the metric property at all.

3. run global segment optimization:
   - embed all segments $\{s_k\}$ into a metric space, $\{\hat{Q}_{s_k}\}$.
   - embed all landmarks $\{l_i\}$ that are observed by more than two segments into the same space, $\{\hat{\boldsymbol{x}}_{l_i}\}$.
   - optimize $\{\hat{Q}_{s_k}\}$ and $\{\hat{\boldsymbol{x}}_{l_i}\}$, into $\{\tilde{Q}_{s_k}\}$ and $\{\tilde{\boldsymbol{x}}_{l_i}\}$.
   - update the map using $\{\tilde{Q}_{s_k}\}$ and $\{\tilde{\boldsymbol{x}}_{l_i}\}$.
4. iterate 1.$\sim$3. until convregence.

In global segment optimization, we iterate the procedure describe above until convergence. In step 3, segments are treated as rigid bodies during optimization. $\hat{Q}_s$ denotes a segment-wise six degree of freedom 3D rigid motion, and the projected pixel coordinate of the landmark l to the keyframe j in the segment k is

$$p(\hat{\boldsymbol{x}}_l, \hat{P}_j, \hat{Q}_k) = K\hat{P}_j\hat{Q}_k\hat{\boldsymbol{x}}_l, \tag{1}$$

where K is the 3×4 camera projection matrix. Note that $\hat{P}_j$ is only updated in Step 2 and kept constant in Step 3. Figure 3 illustrates the proposed global adjustment algorithm.

Since each segment moves as a rigid body, the number of variables in the linear system after Schur complement is $6\times\kappa$. The main idea is to make the global segment adjustment faster by reducing the number of variables, and more stable by grouping nearby keyframes together. As discussed earlier, in the embedded space the inconsistency along a loop is concentrated at the farthest link from the reference keyframe, thus there may be a large opening or overlap around the farthest link. If conventional SBA is used to the individual keyframes for global adjustment, it is likely that the keyframes around this link may not converge to the right pose (see Fig. 5 (b)). However, if segments of keyframes are restricted to move rigidly, contribution from the observations across segment boundaries are accumulated to the segments, and it is more likely to find the correct pose, although the resulting poses of individual keyframes may be slightly less accurate than those from standard bundle adjustment when it converges properly. The small errors that may be caused by rigid segment treatment will be reduced by the within-segment optimization in the next iteration.

As briefly discussed in Section 2, the proposed method has several advantages over existing methods. Using nested dissection with boundary variables [Ni et al., 2007, Ni and Dellaert, 2010] has a serious problem of most of variables being boundary when the graph is not very sparse and segmentation is fine. Long tracks of landmarks provides more information on the keyframes' poses, but at the same time they induce dependencies among all keyframes that commonly observe them, and the sparsity of the entire graph is significantly reduced. The proposed method does not have this issue since it treats each segment as a rigid body (one virtual camera in SBA), so the sparsity structure in the optimization is determined by the relations between the segments instead of the keyframes.

4 Algorithm

In this section we detail our algorithm and its implementation on robotic platforms. The method takes a calibrated stereo video stream as input and generates an environment map of the keyframes and sparse 3D point landmarks. The proposed online environment mapping system consists of four components: Scene Flow Computation, Location Recognition, Local Adjustment and Global Adjustment (see Fig. 4). All components are executed in parallel to minimize the latency and maximize the throughput. We use message passing mechanism to propagate the processing results from a module to other modules.

Scene Flow The Scene Flow module is responsible for detecting and tracking salient features in the input video stream, finding motion inliers among the tracked features and computing the initial estimate of six degree of freedom motion of the camera system. This computation is done on all input video frames, so that the robot can have the pose estimate at all times.

Our feature detection uses the Harris corner detector, limited to the detections on edges [Xiao and Shah, 2003]. This is to ensure that the features are on the true corners in the scene. The detected features are then tracked by two 2D KLT trackers [Lucas and Kanade, 1981] separately running on the left and right video streams. Stereo correspondences of the features are established using Normalized SSD [Brown et al., 2003] when they are initially detected, and it is constantly checked during tracking process if they are on the same epipolar line with

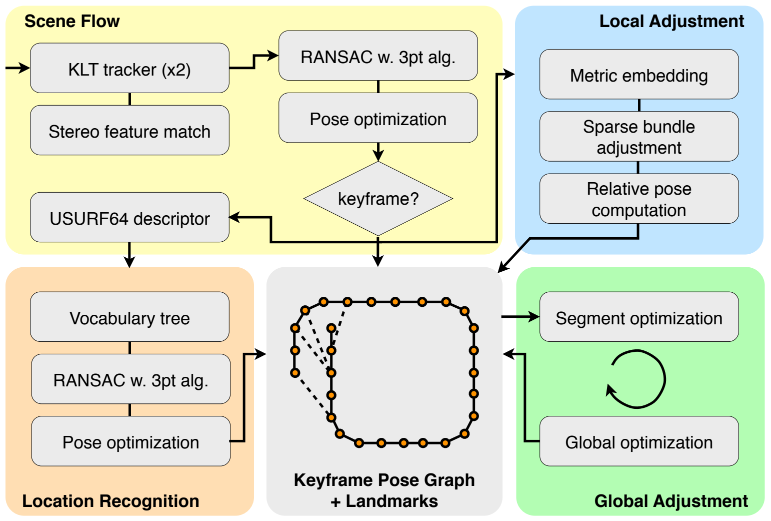


Figure 4: System overview. The system consists of four independently running tasks. Scene Flow processes input stereo frames, tracks features and generates keyframes if there are enough changes. The keyframes are added to the map and passed to Local Adjustment and Location Recognition modules. Location Recognition detects possible loop closure, and Local Adjustment enhances local geometry around the recent keyframes in the map. Global Adjustment periodically checks and optimizes the global map.

valid disparity. The initial 3D position of a landmark is computed using the disparity from the stereo match, and as the camera moves the local and global adjustment processes update the position using all available observations from different view points.

Some of the tracked features may not be locked on fixed scene locations or some may be from independently moving objects. We employ the RANSAC method with 3-point algorithm [Fischler and Bolles, 1997] for robust motion estimation and outlier rejection. Once RANSAC finds the initial 3D camera pose and inlier features, the camera pose is enhanced with a simple non-linear optimization of the six motion parameters using all inliers, and a new set of inliers is found from the updated pose estimate.

New keyframes are created and added to the map when there exist enough camera motion or large change in features. The added keyframes are then passed to the location recognition and the local adjustment modules for further processing. Newly found and tracked features are added as new landmarks and keep updated from incoming observations. The landmarks with too few observations (due to tracking failure or occlusion) are later removed from the map.

To boost the processing speed all low-level image processing routines including image rectification, KLT feature tracking, and stereo feature matching are parallelized and executed on commodity graphics processors.

Location Recognition The location recognition module tries to find loop closures, i.e. if the robot is revisiting a previously-mapped location. Each landmark observation is tagged with USURF-64 [Bay et al., 2008] descriptor, computed on the KLT-tracked feature location in the keyframe. We reuse the feature points for KLT tracking, instead of running separate feature detection. This allows us to save the computation time for extra feature detection, and to ensure that all landmark observations have feature descriptors for matching. Also since the 3D location of each landmark is known, the scale of each feature can be computed from the inverse of the depth to the landmark. The advantages are increased performance by saving the additional computations for scale detection, and better stability in determining the scale of the features. The feature descriptors are computed using integral image technique on the CPU as proposed in [Bay et al., 2008].

For a new keyframe, the location recognition module finds candidate keyframes using the vocabulary tree (width=40, depth=3) [Nister and Stewenius, 2006] for the USURF-64 descriptor. The recent keyframes are excluded from the matching since they share most landmarks with the current keyframe. The vocabulary tree is trained off-line using 1.76 million descriptors from various indoor and outdoor training videos. For the candidate keyframes the relative poses from the new keyframe are computed using RANSAC with the 3-point algorithm similarly to the scene flow module. The candidate with most inliers (above the given threshold) is chosen as the location recognition result, and its relative pose estimate and the inlier set are further improved by a non-linear optimization. If a match (loop closure) is successfully found, a new link connecting the current and the detected keyframe is added into the map. This link will then be optimized by both the local and the global adjustment modules.

Local and Global Adjustment The local adjustment module adjusts the recently added keyframes and landmarks as described in Section 3.2. We use the sparse bundle adjustment algorithm [Lourakis and Argyros, 2009] with pseudo Huber norm which is implemented internally using LAPACK.

The global adjustment module performs the optimization of the entire map as described in Section 3.3. The keyframes, which are currently considered in the local adjustment process are excluded from the global adjustment to avoid inconsistencies by updating the same link in different modules. Within-segment optimization is performed in the same way as the Local Adjustment with all keyframes in the segment as active keyframes. For global segment optimization, we need to compute a Jacobian matrix for segment-wise motion (Eqn. 1), and the rest is similar to the local adjustment.

To make global adjustment use as many keyframes as possible, the global optimization iterates only once and new segmentation is formed using all available keyframes including newly added keyframes after the previous iteration.

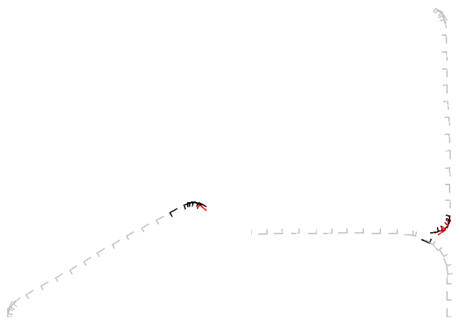
5 Experimental Result

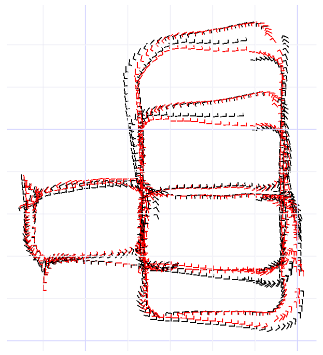
We performed extensive experiments with real video inputs from indoor and outdoor. The test scenarios include various platforms (a humanoid to a car), various moving speed (more than 80 km/h), and many other challenging situations such as dynamic objects, sudden illumination changes, etc. We use three stereo camera rigs for our experiments. The one with 7.5cm baseline and about $110°$ horizontal field of view is mounted in the head of a humanoid robot. The second stereo rig has 16 cm baseline and about $95°$ horizontal field of view and is mounted at the front of an electric cart. The third system with 60 cm baseline is mounted on a car. The effective image resolution after rectification is 640×360 pixel, and the test videos are recorded at $12\sim15$ fps (frames-per-second). No other sensors (such as a GPS or a vehicle odometer) are used in the experiment.

First we demonstrate that our proposed local adjustment can handle the topological changes successfully and instantly. The two images shown in Fig. 5 (a) are the local embeddings before and after the loop closure (from the robot sequence in Fig. 6). Note that the topological changes are reflected in the optimization, where the loop closure creates additional links among keyframes.

With only local adjustment and location recognition, the resulting map is only locally metric. Globally the embedded map may have large openings like the black trajectory in Fig. 5 (b). We ran the traditional bundle adjustment algorithm on this embedded map, shown as the red trajectory in the same figure, and it failed to 'close' the openings and converge to the correct geometry. Fig. 5 (c) shows how the proposed global adjustment works. For each iteration, it segments the map into several pieces (the first and third image, where different colors represent the keyframe segmentation), individual segments are optimized locally, afterwards it aligns the segments jointly with all the shared landmarks (the second and fourth image in the figure). The figure shows the global adjustment can handle this very challenging geometric optimization in just a few iterations. This can be verified in many of the experimental result videos shown below.

We built the system with common and readily available libraries (LAPACK, NVidia's CUDA and standard C++ libraries). All experiments are run on a single laptop computer with a quad-core 2.66GHz Core i7 CPU and a GeForce 330M GPU. Multi-threaded system modules and GPU-accelerated low-level image processing routines make the system fully exploit the available computational resources. The input frames from the recorded test sequences are provided to the system at 20 fps, which is $30\sim60\%$ faster than the original frame rates ($12\sim15$ fps). The scene flow module is able to process them in real-time (50 ms including the idle time as seen in Table 1). The location recognition and local adjustment also run without no delay most of times. The global adjustment module slows down as more keyframes and landmarks are added to the system, but still each


(a)


(b)

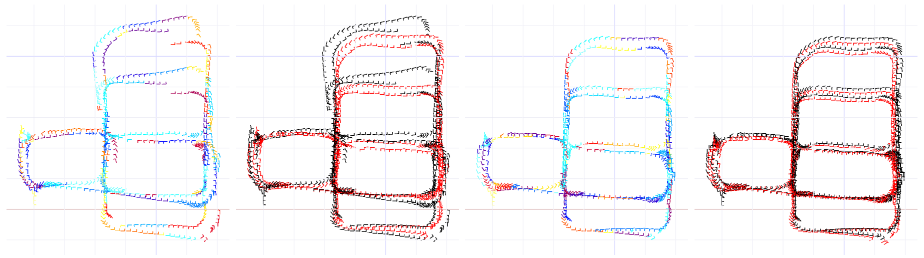

(c)

Figure 5: (a) Local adjustment around a loop closure event. Left image is before a loop closure, and only 5 recent keyframes are active. Right is after the loop closure where additional fixed frames are included in the embedding. (b) Direct application of SBA on an embedding of Fig. 8 without global adjustment. Note that the result is not accurate due to large misalignments in the embedded map. (c) First two iterations of the proposed global adjustment on the same input map as (b) are shown. The first and third images with various colors show the segmentation after segment-wise optimization, and the second and forth show the global segment optimization result. Note that it converges reasonably good even after the first iteration. For all result images, gray is fixed keyframes, black and red are active keyframes before and after the optimization respectively. Only keyframes are shown for presentational clarity, but all associated landmarks are used in optimizations.

iteration runs within 10 seconds in the worst cases at the end of the sequences. The statistics of each module's running time is shown in Table 1, and the modules' actual time consumption can be seen in the timing charts in the figures. For full details on the system's performance, please see the videos in the supplementary material.

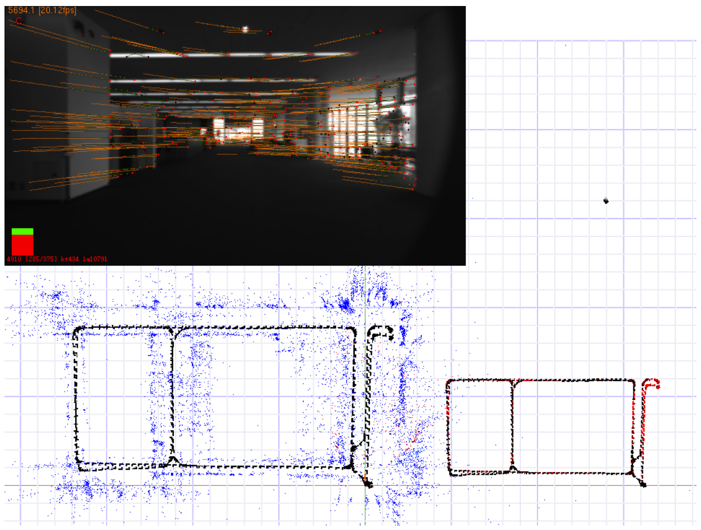
In Fig. 6-8, the top images (marked as (a)) visualize the reconstructed environment map together with the results of global adjustment and local adjustment at the moment. The large map (left-bottom corner) directly under the video frame represents the embedding with respect to the current location (size of the finest grid = 2$\times$2m). In the right-bottom corner (right of the large map), the segmentation and optimization result from global adjustment is shown. Additionally the local adjustment result is displayed in the right side of the video frame to visualize the active and static keyframe sets. In the global and local adjustment results, only keyframes are displayed, and they are marked in different colors per their types: red (before convergence), black (after convergence), and light gray (fixed keyframes).

The bottom images (b) in the figures are the timing charts for the sequences. Three rows visualize the time consumed by each module (except the scene flow module that runs constantly). One pixel in the chart corresponds to 20 ms and one column is one second. In global adjustment row, magenta represents within-segment optimization and black is for global segment optimization. We let the global adjustment run 5 more iterations after the input sequence ends to allow the system generate the final environment map. The main purpose of the timing charts is to see whether the modules are processing the incoming keyframes instantly, or the processing is delayed and the new keyframes are waiting in the queue. In all three experiments, the location recognition and local adjustment runs without delay throughout the sequences, and the global adjustment gets

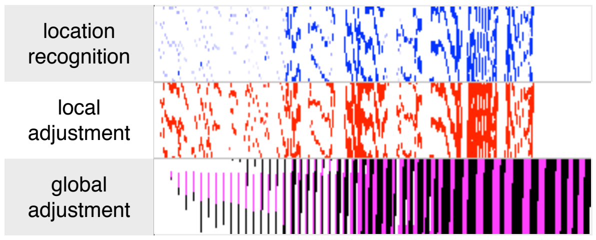
	Robot		Cart-outdoor		Cart-indoor	
Scene Flow	4910×	50.00	4910×	52.00	6900×	52.00
Loc. Rec.	439×	115.49	440×	105.74	636×	158.63
Local Adj.	437×	65.87	430×	136.95	635×	76.38
Global Adj.	46×	2252.81	43×	3501.67	61×	3836.65

Table 1: Average running times (ms) of the modules for each sequence are shown with total number of executions.

delayed as more keyframes and landmarks are added to the map, but it still runs reasonably fast.



(a) the reconstructed environment map at the end of sequence.

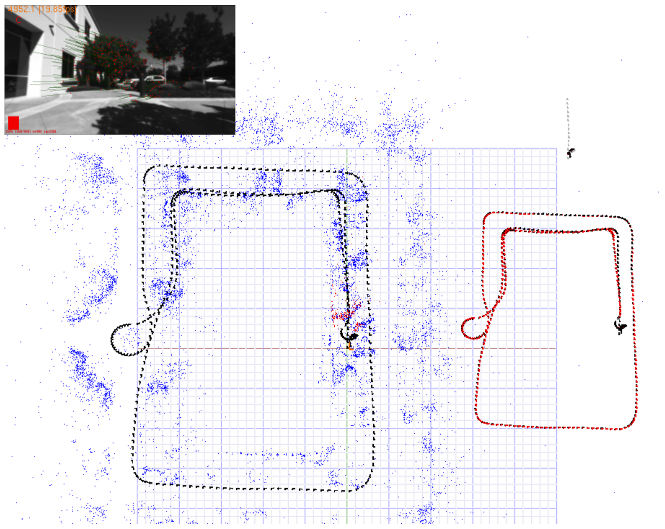


(b) the timing chart for the modules (4910 frames).

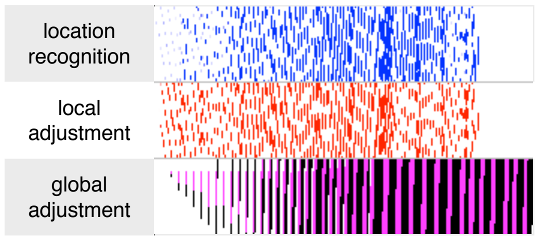
Figure 6: The reconstructed environment map and timing chart from the sequence recorded from a stereo camera on a walking humanoid robot. (a) shows the reconstructed environment map embedded from the current location, together with the results from global and local adjustment, and (b) visualizes the timing information of the modules. Refer the text for further details.

The first sequence (Fig. 6) is from a humanoid robot walking in a large building. Due to the nature of bipedal locomotion and independent head motions, the camera experiences shaking, vibrations, and full 6-DOF motion, but the system was able to robustly track the features in the scene and recover the 3D camera trajectory. In the test sequence there is a corridor with very few features, in which case the motion estimation becomes inaccurate. Despite this locally decreasing accuracy is global adjustment able to deliver the global geometry correctly.

Additionally, we present results of a second indoor scene and an outdoor sequence taken from a moving electric cart. The cart-outdoor sequence (Fig. 7) contains a long travel around a building, and the accumulated motion estimation error is corrected when loops are detected. The cart-indoor sequence (Fig. 8) is also very



(a) the reconstructed environment map at the end of sequence.



(b) the timing chart for the modules (3970 frames).

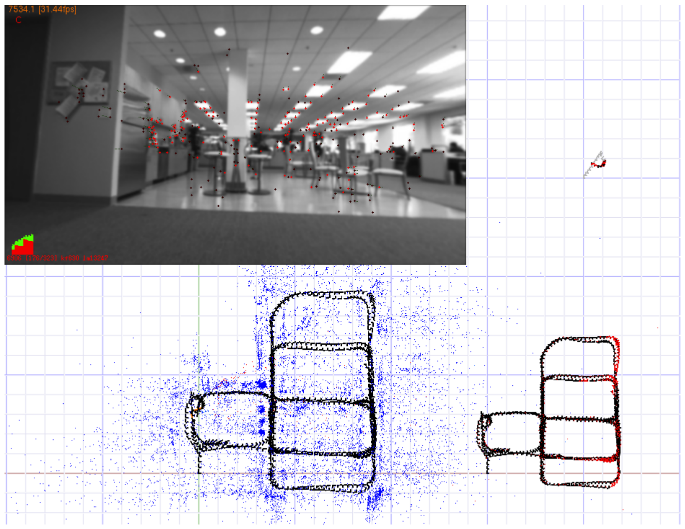
Figure 7: The reconstructed environment map and timing chart from the sequence recorded from a stereo camera mounted on an electric cart running outside of a building. (a) shows the reconstructed environment map embedded from the current location, together with the results from global and local adjustment, and (b) visualizes the timing information of the modules.

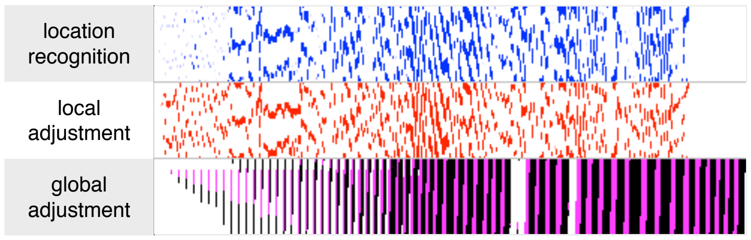
interesting since the distance to the landmarks ranges from very close to far, as well as the sequence contains a large number of loops. Traditional metric mapping methods have difficulties when many loop closures occur within a short time, since there is not enough time to update the map before the next loop closes. Our proposed method keeps the local geometry in the map to be metric and correct all the time, whereas the global metric property is improved as global adjustment progresses.

We also mounted the wider-baseline stereo system on a car and tested the system in many challenging environments. In Fig. 9 $\sim$ 12, each grid in figures represent 20×20 meters. Like previous sequences the three embedded maps are shown in each figure, but in some figures the global and local embeddings are outside of the viewport due to the large size of the map.

First we present a sequence with a very large loop in downtown streets.Fig. 9 (a) and (b) shows the maps just before and after the loop detection. Due to the topological nature of the map, the loop closure happens instantly and the embedding shows the correct and accurate local geometry around the current location. Fig. 9 (c) shows the embedded map from the first global adjustment after the loop detection. The metric inconsistency which cause an large opening in the bottom part of the map is corrected from the global adjustment. Fig. 9 (d) shows the camera trajectory which clearly shows that the car is driving a different lane from the previous run, which verifies the accuracy and effectiveness of the proposed method.

The next test sequence is recorded in a parking lot of a mall. In this sequence many dynamic objects such as pedestrians and other vehicles appears in the test sequence (Fig. **??** (a)). Since the features from those objects are not the majority compared to the features from the static scene, RANSAC process in the scene flow module is


(a) the reconstructed environment map at the end of sequence.



(b) the timing chart for the modules (6906 frames).

Figure 8: The reconstructed environment map and timing chart from the sequence recorded from a stereo camera mounted on an electric cart running in an office environemnt. (a) shows the reconstructed environment map embedded from the current location, together with the results from global and local adjustment, and (b) visualizes the timing information of the modules.

able to filter the independently moving features and compute the ego-motion. Fig. 10 (b) shows that the overall drift of the system is minimal (note that no loop is detected in this sequence), given that the car's trajectory is fairly complex and the features move quite fast in video due to the short distance to the scene.

Another test sequence is taken while we drive the car for a long distance (∼2.3 km) in normal speeds (30∼40 km/h) in urban streets. The system was able to estimate the car's trajectory and the environment map without any problem or delay. We show the reconstructed trajectories from the two sequences overlaid on the satellite images of streets in Fig. 4.

Finally we show one test sequence with partial failures. We drove the car in a highway in 60∼80 km/h, where there are overpasses, ramps, and many other cars running by. Due to lack of large surrounding objects such as buildings or tall trees, the overall number of tracked features are far less than the urban sequences. Moreover the overpasses create very strong illumination changes, which wipes out most features in the scene. When it went under the overpass, the system missed a few keyframes, but quickly resumed processing the following keyframes. However, when the car is on a ramp, there were a few other cars with high textures on the road, and the scene flow module were tracking the features from the cars instead of those from the static background. This caused the estimated motion to be the relative motion to the nearby cars instead of its true motion. Other than these moments, the system was able to successfully process most part of the sequence. Note that the illumination changes can be handled by a gain-adaptive feature tracker [Zach et al., 2008], and the motion outliers can be rejected systematically by incorporating other informations such as a wheel odometer or a GPS.

6 Discussion and Future Work

In this paper we present a novel method that bridges the topological map representation and the metric property in Euclidean maps. While keeping the benefit of an efficient representation of the topological maps that allows instant loop closing, the proposed method tries to enforce the metric property locally and over entire map. As a result, the system is able to maintain the nearby keyframes and landmarks around the current location always accurate and metrically correct. The global map is also optimized to model the environment correctly while maintaining the metric property, and the proposed segment-based iterative optimization is used for efficiency and robustness in global map update.

One interesting future direction is to keep the number of keyframes and landmarks within a manageable size by merging or trimming redundant map entries when the robot stays in or visits the same place many times. This would allow the map size to be proportional to the size of the space and not to the time spent in the space. Also it would be interesting to see if we can do the global adjustment more efficiently by hierarchically segmenting the keyframes and skipping unnecessary within-segment optimizations for static segments.

References

[Agarwal et al., 2009] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building Rome in a day. In *ICCV*. 1

[A.George, 1973] A.George (1973). Nested dissection of a regular finite element mesh. In *SIAM Journal on Numerical Analysis*, volume 10, pages 345–363. 3

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Surf: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359. 8

[Blanco et al., 2008] Blanco, J.-L., Fernández-Madrigal, J.-A., and González, J. (2008). Towards a unified bayesian approach to hybrid metric-topological slam. *IEEE Transactions on Robotics*, 24(2):259–270. 3

[Brown, 1976] Brown, D. C. (1976). The bundle adjsutment - progress and prospects. In *Archives of Photogrammetry*, volume 21. 3

[Brown et al., 2003] Brown, M. Z., Burschka, D., and Hager, G. D. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(8):993–1008. 7

[Chris Engels, 2006] Chris Engels, Henrik Stewenius, D. N. (2006). Bundle adjustment rules. In *PCV*. 2

[Clipp et al., 2010] Clipp, B., Lim, J., Frahm, J.-M., and Pollefeys, M. (2010). Parallel, real-time visual slam. In *IROS*. 1, 3, 4

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *TPAMI*, 29(6):1052–1067. 1

[Fischler and Bolles, 1997] Fischler, M. A. and Bolles, R. C. (1997). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *IJCV*, 22(2):125–140. 8

[Frahm et al., 2010] Frahm, J.-M., Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. (2010). Building Rome in a Cloudless Day. In *ECCV*. 1

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition. 3

[Holmes et al., 2009] Holmes, S., Sibley, G., Klein, G., and Murray, D. W. (2009). A relative frame representation for fixed-time bundle adjustment in sfm. In *IEEE International Conference on Robotics and Automation*, pages 2631–2636. 3

[Konolige and Agrawal, 2008] Konolige, K. and Agrawal, M. (2008). Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077. 1, 3

[Lipton and Tarjan, 1979] Lipton, R. and Tarjan, R. (1979). Generalized nested dissection. *SIAM Journal on Applied Mathematics*, 16(2):346–358. 3

[Lourakis and Argyros, 2009] Lourakis, M. I. A. and Argyros, A. A. (2009). Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Software*, 36(1):1–30. 9

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 1151–1156. 7

[Ni and Dellaert, 2010] Ni, K. and Dellaert, F. (2010). Multi-level submap based slam using nested dissection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 3, 7

[Ni et al., 2007] Ni, K., Steedly, D., and Dellaert, F. (2007). Out-of-core bundle adjustment for large-scale 3d reconstruction. In *ICCV*. 3, 7

[Nistér et al., 2006] Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1). 2

[Nister and Stewenius, 2006] Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*. 8

[Pollefeys et al., 2004] Pollefeys, M., Gool, L. V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., and Koch, R. (2004). Visual modeling with a hand-held camera. *IJCV*. 2

[Sibley et al., 2009] Sibley, G., Mei, C., Reid, I., and Newman, P. (2009). Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems*, Seattle, USA. 6

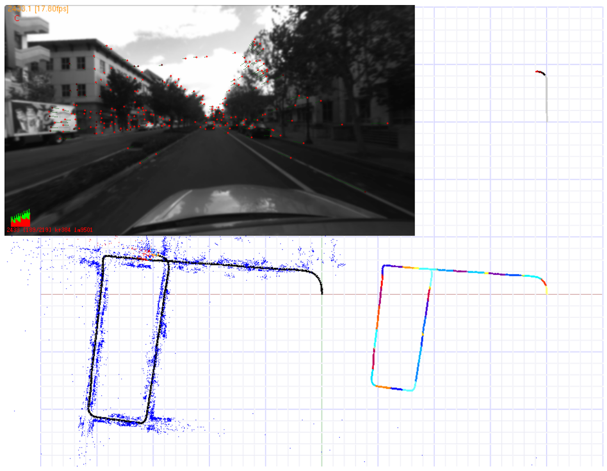
[Snavely et al., 2006] Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, pages 835–846. 2

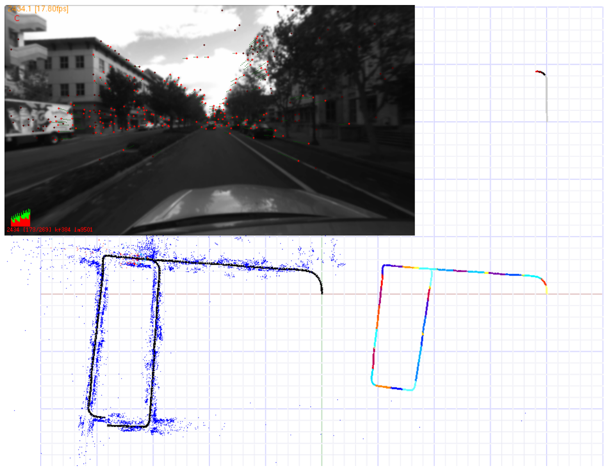
[Snavely et al., 2008] Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Skeletal sets for efficient structure from motion. In *CVPR*. 3

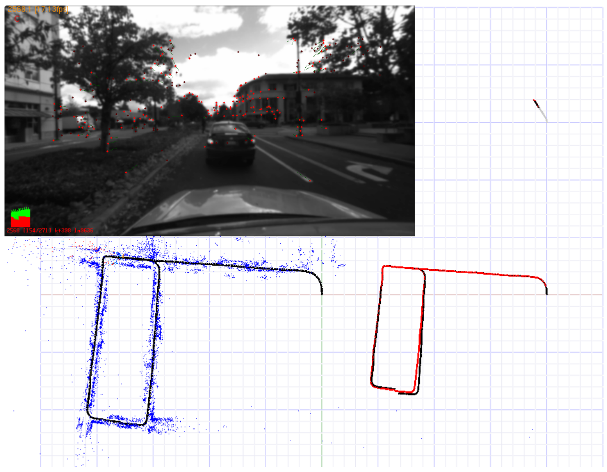
[Triggs et al., 2000] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle Adjustment A Modern Synthesis. In Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *LNCS*, pages 298–372. Proc. of the Intl. Workshop on Vision Algorithms: Theory and Practice, Springer-Verlag. 2

[Xiao and Shah, 2003] Xiao, J. and Shah, M. (2003). Two-frame wide baseline matching. In *In Proceedings of the International Conference on Computer Vision*, pages 603–609. 7

[Zach et al., 2008] Zach, C., Gallup, D., and Frahm, J.-M. (2008). Fast gain-adaptive klt tracking on the gpu. In *CVPR Workshop on Visual Computer Vision on GPU's (CVGPU)*. 13


(a) before loop-closure.


(b) after loop-closure.


(c) first Global Adjustment after the loop closure.

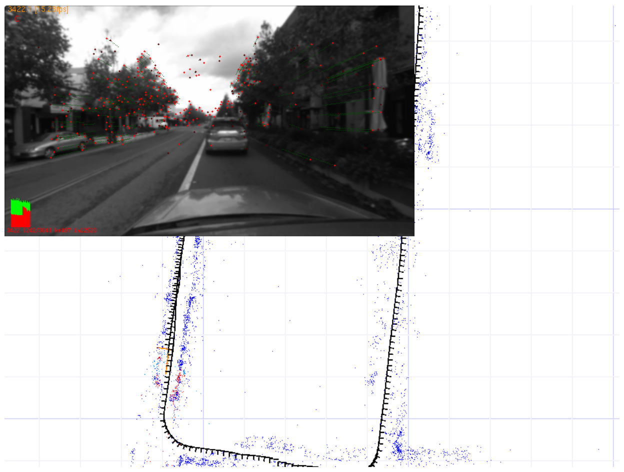
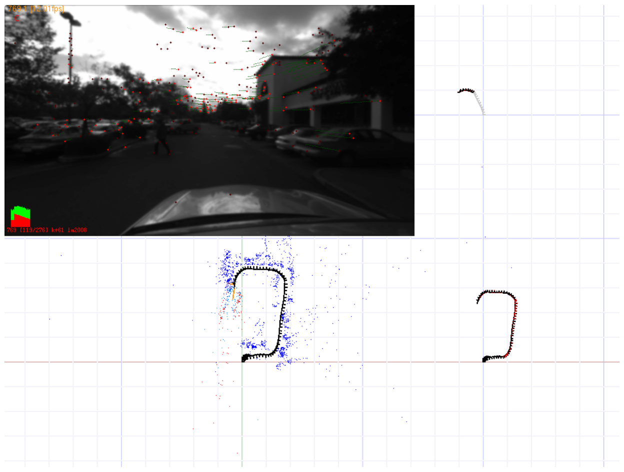

(d) driving a different lane.

Figure 9: The snapshots from a car sequence 'Castro St.' (a, b) The first loop closure occurs when the car merges into the street that it has driven before. Note the embedded map shows the broken link at the farthest location from the current position. (c) After the Global Adjustment is done, the global metric consistency is recovered and the embedded map shows a smooth loop. (d) As the car drives the same street, small loop closures repeatedly occurs. From the zoomed-in view, one can see that the car is driving a different lane from the previous run.


(a) dynamic objects in the scene.

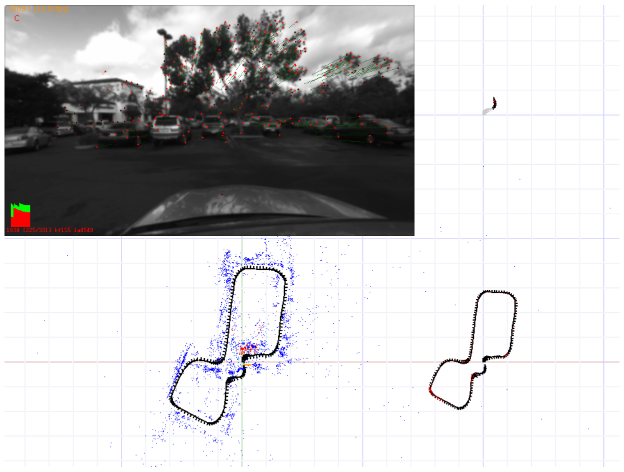
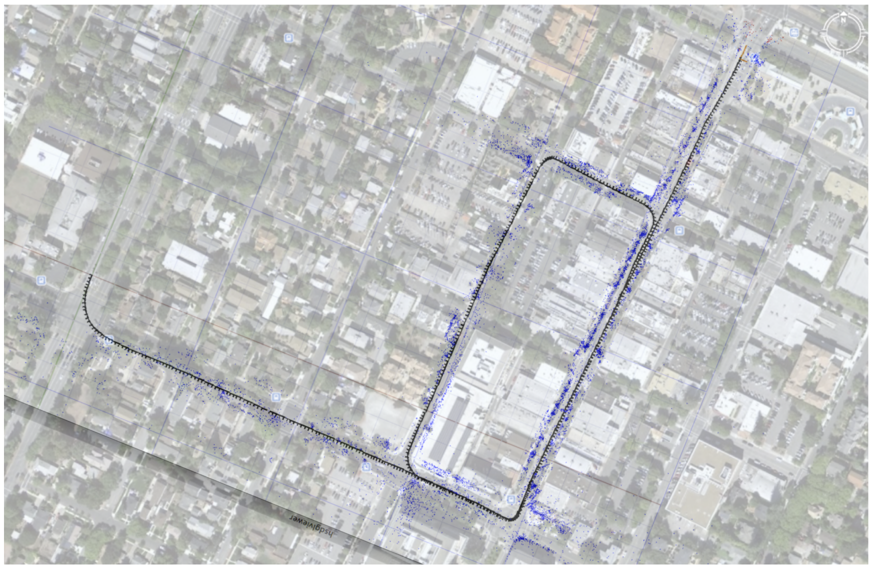

(b) returning to the starting position.

Figure 10: The snapshots from a car sequence 'Parking Lot'. (a) The features from moving objects (a pedestrian and a truck) are regarded as outliers in RANSAC process, and not included in the map. (b) After a loop in a parking lot, the car's location is quite accurately estimated even though the loop closure from feature matches did not occur.


(a) 'Castro St.'

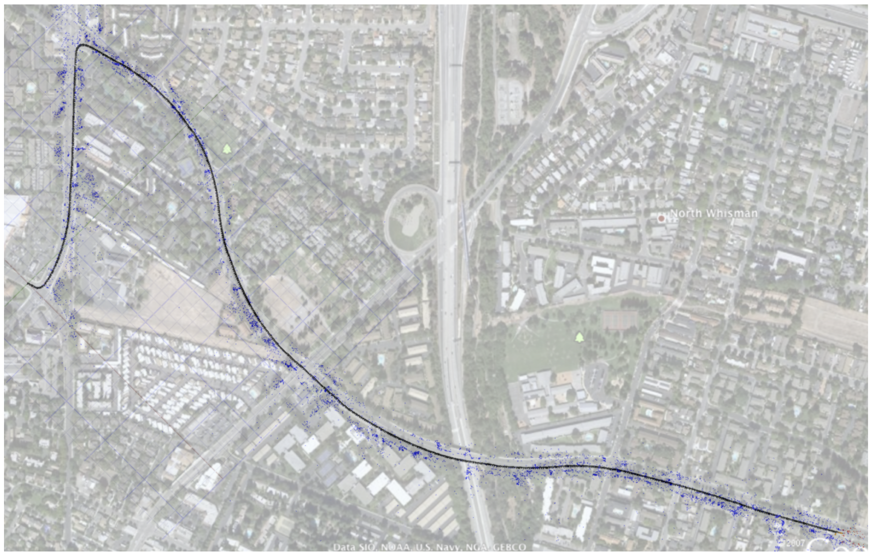
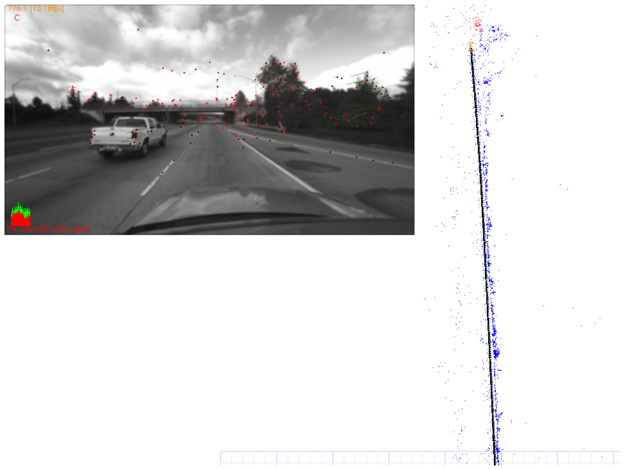
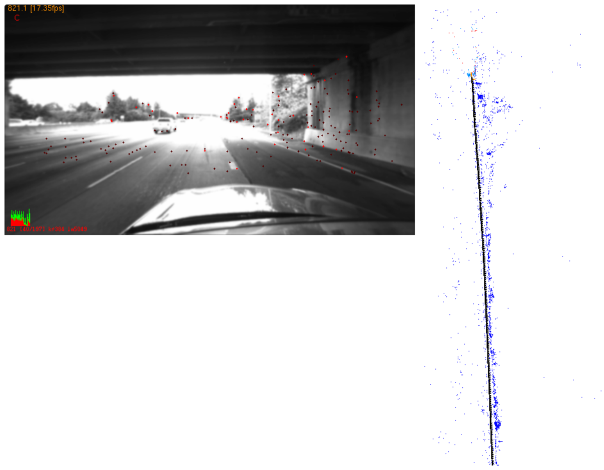
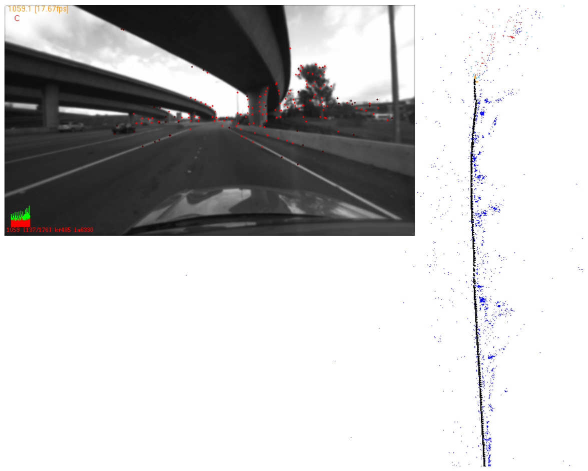

(b) 'Middlefield Rd.'

Figure 11: The environment maps are overlaid on top of the aerial photos from Google Earth. The global location, rotation, and scale of the map plots are manually adjusted to the aerial photos. The results show very high accuracy in spite of the long distances travelled.


(a) successful runs.


(b) drastic illumination change.


(c) Global Adjustment error.

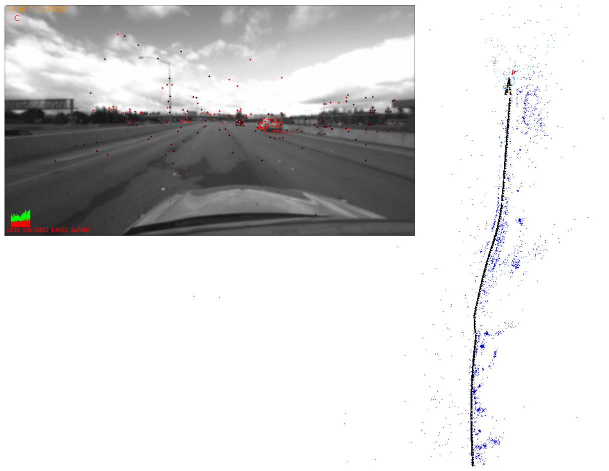

(d) dominant features from moving objects.

Figure 12: The snapshots from a car sequence ’Highway 101’ (a) The car drives high speed about 80 km/h in an urban highway, but the system can successfully conduct mapping and localization most of the time. (b) Due to sudden illumination condition changes, feature tracker was not able to track enough features. (c) Excessive amount of false feature matches made the Global Adjust converge to a wrong pose. (d) Feature tracker latches on the highly textured cars nearby, and the localizer fails to estimate correct ego-motion.