Visual SLAM with a Multi-Camera Rig

Michael Kaess and Frank Dellaert
College of Computing
Georgia Institute of Technology
{kaess,dellaert}@cc.gatech.edu
Technical Report GIT-GVU-06-06
February 2006

Abstract

Camera-based simultaneous localization and mapping or visual SLAM has received much attention recently. Typically single cameras, multiple cameras in a stereo setup or omni-directional cameras are used. We propose a different approach, where multiple cameras can be mounted on a robot in an arbitrary configuration. Allowing the cameras to face in different directions yields better constraints than single cameras or stereo setups can provide, simplifying the reconstruction of large-scale environments. And in contrast to omni-directional sensors, the available resolution can be focused on areas of interest depending on the application. We describe a sparse SLAM approach that is suitable for real-time reconstruction from such multi-camera configurations. We have implemented the system and show experimental results in a large-scale environment, using a custom made eight-camera rig.

I. Introduction

Camera-based simultaneous localization and mapping (SLAM) has received much attention recently. Cameras are small and cheap sensors with low power consumption, which makes them suitable for large scale employment in commercial products such as robotic toys and home appliances. Traditional offline approaches from structure from motion (SFM) research in computer vision have been available for quite some time now [7], but for many applications offline approaches are not acceptable. More recently some real-time systems were introduced for visual odometry [13], which only recovers the camera trajectory and does not explicitly reconstruct a map, as well as for visual SLAM [15], [2], [9]. However, all approaches are either based on a single camera [2], [13], [9], on multiple cameras in a stereo configuration [15], [13], where all cameras face in the same direction, or on omni-directional cameras [11]. No large scale visual SLAM results have been presented so far. One reason is the restricted field of view of single cameras and traditional stereo setups, that do not provide enough constraints for large scale environments. Even though omni-directional cameras provide a large field of view, they suffer from an unfavorable spatial distribution of the available resolution.

We propose to use a multi-camera setup that is not in any specific stereo configuration, but where the cameras face in different directions. Such a *multi-camera rig* combines the advantages of omni-directional vision with those of single cameras. In contrast to single cameras and traditional stereo setups, a multi-camera rig covers a wider field of view, leading to better constraints for localization. And in contrast to omni-directional cameras, that distribute the available pixels over the complete scene, a multi-camera rig can focus the available resources

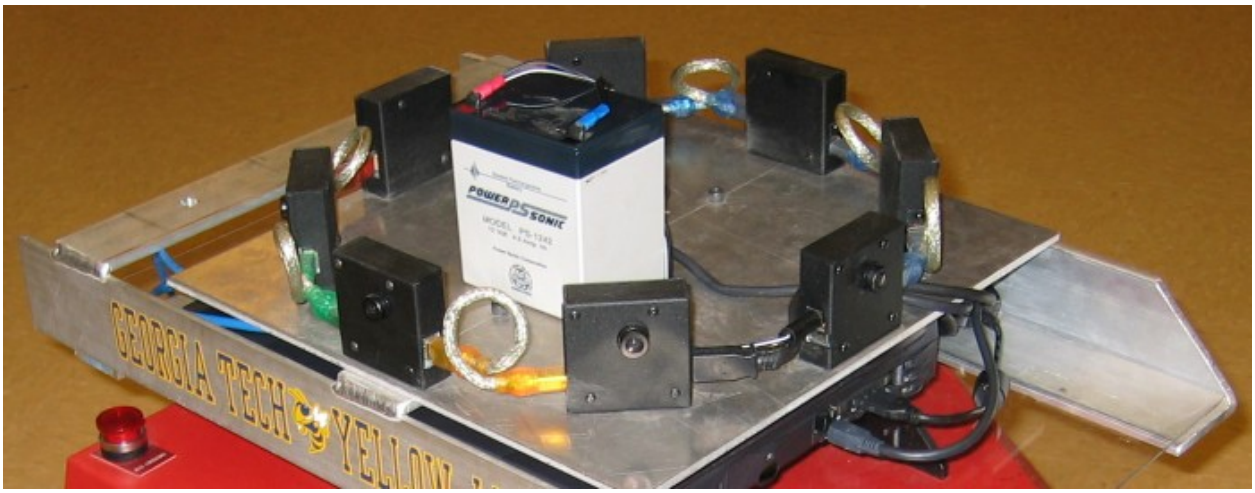

Fig. 1. Custom made 8-camera rig, mounted on top of an ATRV-Mini mobile robot platform. The FireWire cameras are distributed equally along a circle and connected to an on-board laptop.

on areas of interest depending on the application. And similar to stereo setups, a multi-camera rig supports metric reconstructions, unlike single omni-directional sensors.

We present an algorithm to perform simultaneous localization and mapping with a multi-camera rig in constant time. No specific assumptions on the configuration of the rig, like overlapping views or equal spacing of the cameras, are made, so that an advantageous configuration for a specific application can be chosen, while obeying any potential constraints. In particular it is not necessary to place all cameras in a central location. Furthermore, including back facing cameras not only provides better constraints, but also allows the robot to drive backwards, which is essential for wheel based robots, but is not addressed by most current methods. A more general configuration, as we will use for this work, is shown in Fig. 1, where the cameras are distributed equally along a circle.

In terms of related work, multi-camera rigs appear in the literature in the context of image-based rendering and SFM. A theoretical treatment of multi-camera systems in SFM is presented in [14]. Localization with one or more cameras is presented in [12]. Levin [10] uses the Point Grey Ladybug six-camera rig in combination with a hand drawn map for offline loop closing in the context of visual odometry, which does not create a map. In terms of the underlying SLAM algorithm, our work is closely related to the large field of SFM in computer vision [7]. Furthermore, there is some related work on visual SLAM [15], [2], [3], [9]. Se [15] uses a trinocular stereo sensor to recover depth of SIFT features. The recovered structure is integrated over time into sub-maps that are assumed to be consistent by themselves, and which can be aligned with respect to each other. Results from inside a laboratory are shown. Davison [2] presents single-camera mapping and localization in real-time at a high frame rate, but restricted to a desk-like environment. In an extension in [3], the benefits of using a single wide-angle camera over a normal camera are shown. [9] uses a single camera to generate landmarks as 3D structure of special locations in a home environment. Nister [13] tracks a single or stereo camera in real time at a high frame rate to obtain visual odometry, without explicitly creating a map.

II. Localization and Mapping

We want to find the map and trajectory that best explain the image and odometry measurements of the robot. This corresponds to the structure from motion problem, but additionally odometry is available as a further constraint. The robot's trajectory, or its motion, $M = \{\mathbf{m}_i\}_{i=1}^{m}$ specifies its pose $\mathbf{m}_i$ at each time i. Each robot pose $\mathbf{m} = (R, \mathbf{t})$ consists of a 3D translation $\mathbf{t}$ and a 3D rotation R, that can be specified by the three Euler angles yaw ϕ, pitch θ and roll ψ. The odometry $O = \{\mathbf{o}_i\}_{i=1}^{m-1}$ consists of measurements $\mathbf{o}_i$ of the difference in pose between subsequent steps. The map $X = \{\mathbf{x}_j\}_{j=1}^{n}$ contains 3D translations $\mathbf{x}_j$ for certain points in the environment that we can observe through the visual input V. The problem can now be formulated as finding the best map X and trajectory M given some visual input V and odometry O. This can be transformed via Bayes law to a likelihood term $L(X, M; V, O) \propto P(V, O|X, M)$ that can be defined by a generative model, and a prior $P(X, M)$ that can be based on the odometry. This is equivalent to the minimization of the sum of the log-likelihoods:

$$\begin{aligned} X^*, M^* &= \underset{X,M}{\text{argmax}} \; P(X, M|V, O) \\ &= \underset{X,M}{\text{argmax}} \; L(X, M|V, O)P(X, M) \\ &= \underset{X,M}{\text{argmin}} \; (\log L(X, M; V, O) + \log P(X, M)) \end{aligned} \tag{1}$$

A. Camera Rig Projections

Before we can define a generative model for the log-likelihood $L(X, M; V, O)$ as needed in (1), we first have to take a closer look at how images are generated by a multi-camera rig. A *multi-camera rig* is a set of c cameras fixed with respect to each other and the robot. Note that we make no assumptions about the orientation of the cameras. They can face in any direction that seems suitable for a specific application, as long as they do not move with respect to each other and the robot. An image set is obtained by synchronously acquiring an image from each camera. It can be thought of as a single *joint image*, where the pixel $\mathbf{p} = (u, v)$ in rig camera r is addressed by the tuple $(r, \mathbf{p})$. Note that this corresponds to a single camera with multiple optical centers, ie. a single point in the environment can have more than one projection in the joint image.

Projecting a 3D world point $\mathbf{x}$ into a rig camera is a two-step process. First, the world point is transformed to rig coordinates $\mathbf{x}'_i = R_i^T(\mathbf{x} - \mathbf{t}_i)$, where the 3D rotation R_i and translation $\mathbf{t}_i$ determine the pose of the camera rig at time i. Second, this point $\mathbf{x}'_i$ is projected into rig camera $r \in \{1...c\}$ using standard methods [7]. The overall projection $\Pi_{i,r}(\mathbf{x})$ is given by

$$\Pi_{i,r}(\mathbf{x}) = K_r[R_r|\mathbf{t}_r]R_i^T(\mathbf{x} - \mathbf{t}_i) \tag{2}$$

with

$$K = \begin{bmatrix} \alpha_x & s & u_0 \\ & \alpha_y & v_0 \\ & & 1 \end{bmatrix}$$

where K contains the *intrinsic* calibration parameters of the camera, that usually include the two focal lengths α_x and α_y in terms of pixel dimensions, the skew s, and the principal point

Fig. 2. An example of radial distortion removal, comparing the original image (left) with the corrected image (right).

$\mathbf{p}_0 = (u_0, v_0)$. The *extrinsic* calibration parameters consist of the 3×3 rotation matrix R and the translation vector $\mathbf{t}$ of the camera with respect to the center of the robot. The overall rig calibration can then be summarized as $\{K_r, R_r, \mathbf{t}_r\}_{r=1}^{c}$ and can be determined in advance or by automatic calibration.

Additionally it is often necessary to model *radial distortion*. We approximate radial distortion by the quadratic function $r_D = r + \kappa r^2$ with a single parameter κ, as well as the center of distortion $\mathbf{p}_D = (u_D, v_D)$ that might be different from the principal point $\mathbf{p}_0$. r and r_D are the radii of the projected point in ideal coordinates before and after distortion, respectively. To remove radial distortion, the incoming images are warped efficiently using a look-up table that is calculated only once. Fig. 2 shows an example image with significant radial distortion and its corrected counterpart.

B. Trajectory and Map

We can now define the log-likelihood from (1) in terms of generative models for image and odometry measurements, in order to find the best trajectory and map given the measurements. The image measurement $\mathbf{p}_{i,r} = \Pi_{i,r}(\mathbf{x}) + \mathbf{v}$ of 3D point $\mathbf{x}$ in rig camera r at time i is the sum of the geometric prediction $\Pi_{i,r}(\mathbf{x})$ from (2) and measurement noise $\mathbf{v}$, that we assume to be i.i.d. zero-mean Gaussian $\mathbf{v} = \mathcal{N}(\mathbf{p}; 0, \Sigma)$. Similarly, the odometry measurement $\mathbf{o}_i = \delta(\mathbf{m}_i, \mathbf{m}_{i+1}) + \mathbf{w}$ is given by the difference $\delta(\mathbf{m}_i, \mathbf{m}_{i+1})$ of two successive poses $\mathbf{m}_i$ and $\mathbf{m}_{i+1}$ with measurement noise $\mathbf{w} = \mathcal{N}(\mathbf{o}; 0, \Xi)$ added. Assuming known correspondences $T = \{(i_k, j_k, (r_k, \mathbf{p}_k))\}_{k=1}^{l}$, where the triple $T_k = (i_k, j_k, (r_k, \mathbf{p}_k))$ describes an image measurement $\mathbf{p}_k$ of map point j_k as observed by rig camera r_k at time i_k, equation (1) simplifies to the minimization of a sum of terms over l correspondences and $m - 1$ odometry measurements

$$\sum_{k=1}^{l} ||\mathbf{p}_k - \Pi_{i_k, r_k}(\mathbf{x}_{j_k})||^2_{\Sigma_k} + \sum_{i=1}^{m-1} ||\mathbf{o}_i - \delta(\mathbf{m}_i, \mathbf{m}_{i+1})||^2_{\Xi_i}$$

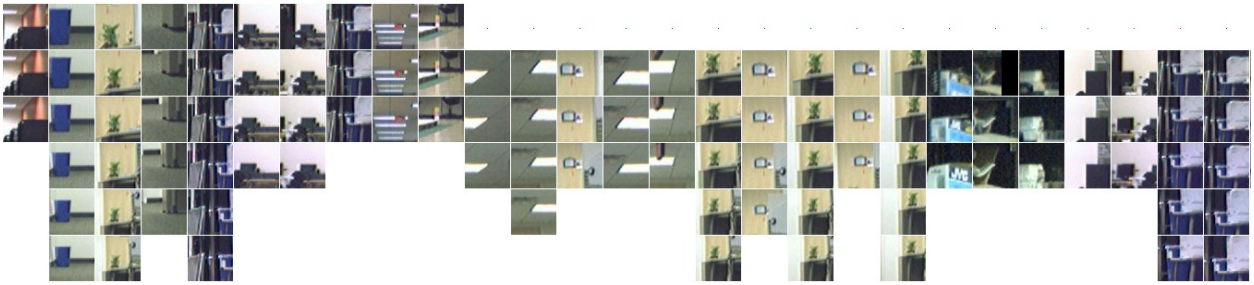

Fig. 3. Correspondences are shown as templates of the measurements of different points (columns) over time (rows). To provide more context in the visualization, much larger templates are shown than the ones that are actually used in the matching process. Note that objects appear larger or smaller over time depending on the motion of the robot and the bearing at which the feature is observed.

where $||\mathbf{y}||^2_\Sigma = \mathbf{y}^T\Sigma^{-1}\mathbf{y}$. We apply Levenberg-Marquardt, a non-linear minimization algorithm that can achieve fast convergence to the global minimum based on a good initial estimate obtained from odometry and triangulation.

A real-time application will not be able to optimize the complete map and trajectory after each new measurement is incorporated. Instead, we use fixed-lag smoothing, that only optimizes over a constant number of steps, an operation with complexity constant in the size of the map and the length of the trajectory. The SLAM literature provides many other approximation algorithms that can be used here.

C. Correspondences

So far we have assumed known correspondences between image measurements. We will now explain how to obtain them automatically from the data. We first identify interest points in the incoming images, then generate putative matches between different images, and finally remove any outliers to obtain a set of correct correspondences.

To *identify features* in the input images, we use the Harris corner detector [6]. After thresholding on the Harris response, we perform non-maximum suppression, selecting the best feature in any 5×5 neighborhood. To prevent an overly large number of features in highly textured environments, only a limited number of features corresponding to the highest Harris responses are accepted.

To *generate putative matches*, we compare features based on the appearance of their local neighborhood. We define a template $\mathbf{v}$ to be the vector of intensity values of a square region centered on the feature. The dissimilarity d_{ab} of two templates $\mathbf{v}_a$ and $\mathbf{v}_b$ is given by their sum of square differences $d_{ab} = (\mathbf{v}_a - \mathbf{v}_b)^T(\mathbf{v}_a - \mathbf{v}_b)$. To match the features of two images A and B we employ the mutual consistency check. For each feature in image A we select the feature of image B that minimizes the dissimilarity. We repeat this step with A and B swapped. The matches that agree in both directions are putative matches. Note that, for a camera rig, A and B are joint images that each consist of multiple real images. Matches cannot only occur between images taken by the same rig camera over time, but can also stretch across different rig cameras. To make the matching process more efficient, we restrict the search region to some area around the odometry-based epipolar prediction.

Typically, the resulting set of putatives still contains wrong matches, especially in the presence of repetitive textures. To find the correct *correspondences* from the putative matches, a random sampling based algorithm like RANSAC [4] can be used in combination with a geometric constraint, like the fundamental matrix between image pairs. A more robust constraint [1] is provided by the trifocal tensor, where the features have to geometrically agree over three images. To obtain the necessary putative matches across triples of images (A, B, C), we combine the pairwise matches of $A - B$, $B - C$, and $A - C$ by again using the mutual consistency check. RANSAC is implemented according to [7]. For a minimal sample of putative matches we need to find the corresponding camera poses. A DLT solution using Singular Value Decomposition (SVD) is not directly available for the multi-camera rig. Instead we apply bundle adjustment, which performs equally well when implemented correctly [16]. The inlier check is based on the residual error for a putative match. The inliers either overlap with an existing feature track, in which case the track is extended by one, or they define a new track, in which case a new point is added to the map. Fig. 3 shows an example of correspondences extracted by our method.

D. Summary

Our algorithm as described in detail above, can be summarized in terms of a pipeline, which is executed for each incoming joint image:

1) Radial distortion removal
2) Feature detection: Harris, non-maximum suppression, limit number
3) Trifocal matching across cameras and time: SSD, mutual consistency, epipolar prediction to restrict search region
4) Robust outlier removal: RANSAC over three time steps
5) Map and trajectory update: fixed-lag smoothing

III. RESULTS

We have implemented and tested our method using a custom made eight-camera rig. The cameras are evenly distributed along a circle on the rig, facing outwards as shown in Fig. 1. The rig is mounted on top of an ATRV-Mini mobile robot platform and connected to an on-board laptop. Each camera has a field of view of slightly over 45 degrees, yielding complete coverage of the surroundings of the robot. We use cheap FireWire cameras that allow streaming of 640×480 color images at up to 7.5 fps for all cameras simultaneously. The robot additionally provides odometry data. For efficient computation, we use sparse matrix operations to allow for fast non-linear optimization of systems with hundreds of variables. An automatic differentiation framework [5] allows us to calculate a Jacobian at any particular given value, efficiently, and free of numerical instabilities.

We have driven our robot in a double loop through the hallways of the second floor of the Technology Square Research Building at Georgia Tech. The bounding box of the robot trajectory is about $30m$ by $50m$. 260 joint images were taken with variable distances of up to 2 meters between successive views. We perform fixed-lag smoothing with a lag of 10. All processing takes less than 5 seconds per joint image (Pentium M 2GHz laptop) and is constant in the size of the map. Even though 3DOF seems sufficient for this office building environment, it turned out that

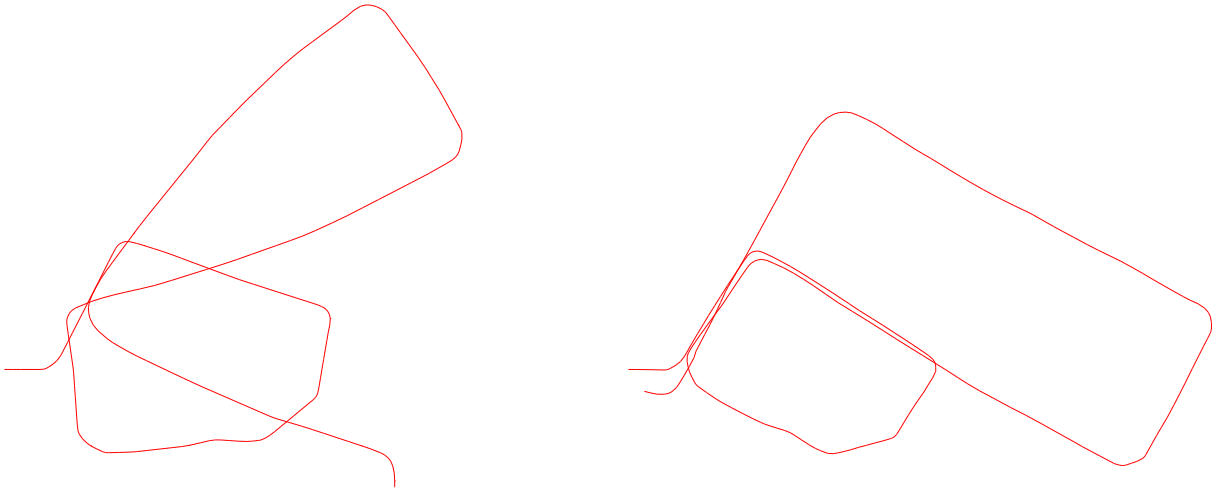

Fig. 4. Left: Odometry as provided by the robot platform. Right: Trajectory as reconstructed by our approach using visual input and odometry only.

6DOF with a prior on pitch, roll and height is necessary, since any small bump in the floor has a clearly visible effect on the images. The standard deviations on x and y are $0.1m$ and on ϕ (yaw) $0.2rad$. The priors on z, θ (pitch) and ψ (roll) are all 0 with standard deviations $0.01m$, $0.02rad$, $0.02rad$ respectively. Only 3DOF odometry was available from the robot's wheel encoders.

On average 320 features were detected per image, of which 68 got matched over time, with an average track length of 4.06. The final map consists of 4383 points constraint by 17780 measurements. The odometry as measured by the robot is shown in Fig. 4, together with the trajectory as output by our system based on visual input and odometry only. The same trajectory together with the map of sparse features is shown in Fig. 5, manually overlayed with the floor plan for comparison. The same sparse 3D map is shown from the side in Fig. 6, while Fig. 7 provides an inside view of the map.

The localization error after a trajectory length of 190 meters is about one meter (note that the actual start and end points are not the same for this data set). The algorithm does not perform explicit loop closing, as only incremental matching is applied. Loop closing could be performed by additionally matching against previous frames that are geometrically close in the current estimate. However, this adds computation and requires optimization along the complete loop, resulting in a time complexity that is not independent of the map size.

One reason for the short average track length is the pairwise matching between three images, and especially the mutual exclusion constraint applied to each pair. A similar feature in one of the three images can prevent a correct correspondence from being considered as a putative match. The use of SSD rather than a more viewpoint invariant matching criterion also plays a roll. However, we observed that SSD on small templates is a reasonably good metric for our successive matching process, since changes in view point are limited.

Fig. 5. Trajectory and map as reconstructed by our approach using visual input and odometry only. Each robot pose is shown as the projection of the axes of the local 3D coordinate system (X axis in red facing forward, Y axis in green to the left, Z axis in blue facing up). The sparse structure is shown as black points. For comparison the manually aligned building map is shown in gray. Even though the trajectory does not line up exactly, it is very close considering the large scale of the environment and the incremental nature of the reconstruction method.

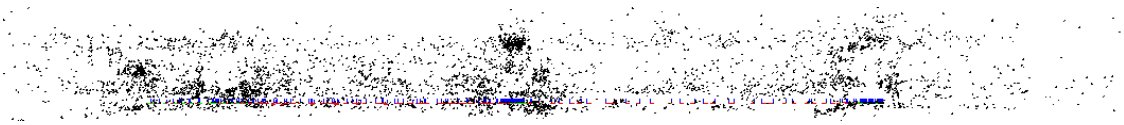

Fig. 6. A side view of the 3D reconstruction using the same colors as in Fig. 5. Note that we do not make any assumptions about the environment, like the presence of flat walls or a limited height of rooms.

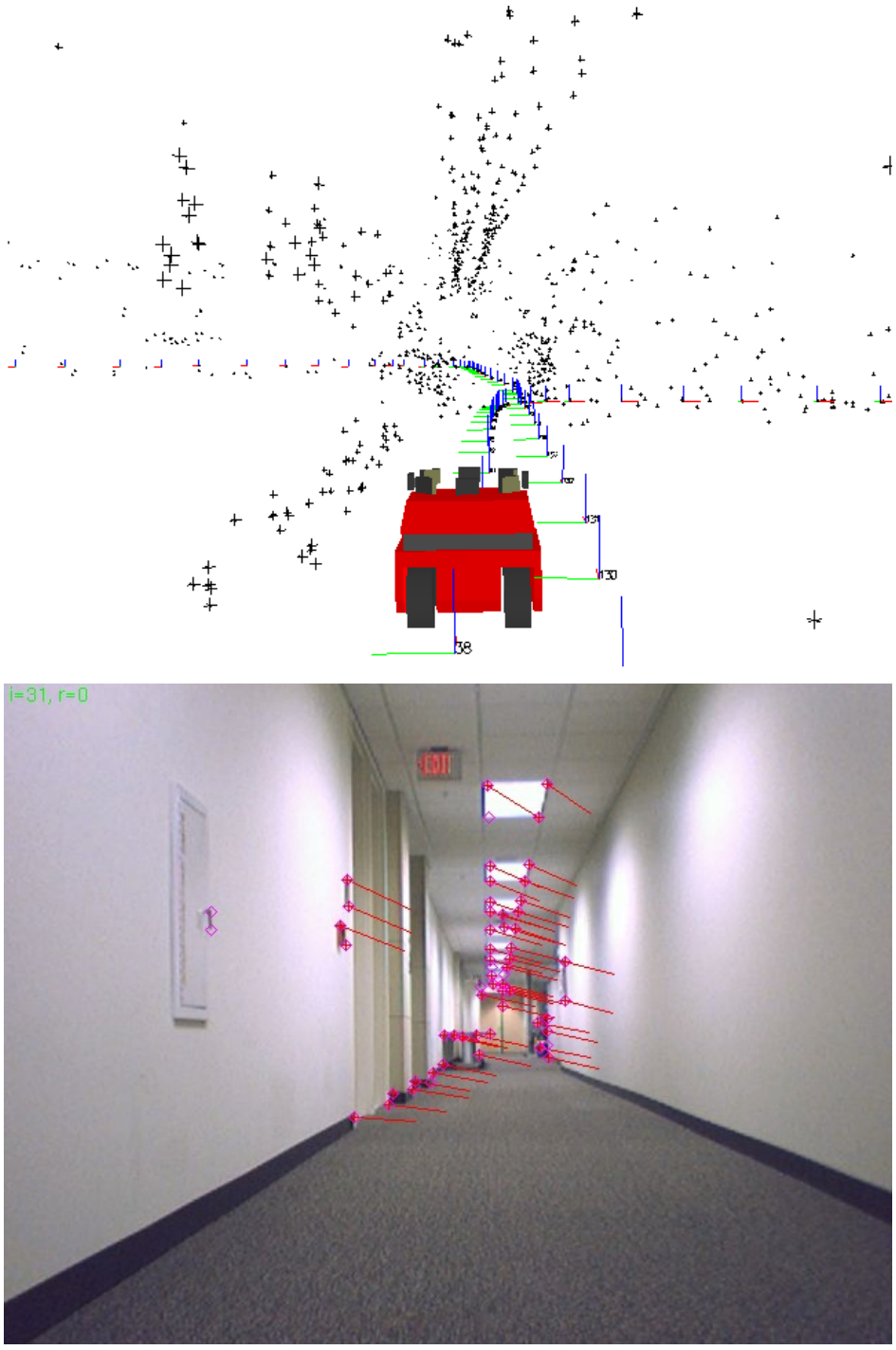

Fig. 7. Comparison of 3D model and actual image. Top: The 3D map as seen from inside a corridor showing the structure (black crosses), the robot coordinate systems (red,green,blue axes for x,y,z) and a 3D model of the robot. Bottom: A real image for comparison, showing the features (magenta diamonds), image measurements (red crosses) and optical flow (red lines). In the 3D view, the lamps along the ceiling can be seen twice with a slight shift, since the robot traversed the corridor twice, and no loop closing is performed. Note that the error is small, given the large scale of the environment.

IV. Conclusion

We have presented a novel approach to camera-based SLAM with multiple cameras in a general, non-stereo setting. We have implemented and successfully tested the system in a large-scale environment using a custom made 8-camera rig. Even though no explicit loop closing is performed at this time because of the incremental nature of the algorithm, the reconstructed trajectory is topologically correct, and metrically close to the correct solution. Our results exceed previous work in visual SLAM in terms of the size of the environment and quality of the reconstruction. The algorithm already runs in time constant in map size and trajectory length, but further improvements are necessary to achieve real-time on a single on-board computer. The main limitation of this work is the lack of explicit loop closing, which is subject to future work, where we plan to extend our previous laser based approach [8] to the visual domain.

References

[1] P.A. Beardsley, P.H.S. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *Eur. Conf. on Computer Vision (ECCV)*, pages II:683–695, 1996.

[2] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Intl. Conf. on Computer Vision (ICCV)*, pages 1403–1410, 2003.

[3] A.J. Davison, Y.G. Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[4] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[5] A. Griewank. On Automatic Differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, 1989.

[6] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, August 1988.

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

[8] M. Kaess and F. Dellaert. A Markov chain Monte Carlo approach to closing the loop in SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 645–650, Barcelona, Spain, April 2005.

[9] N. Karlsson, E.D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M.E. Munich. The vSLAM algorithm for robust localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 24–29, 2005.

[10] A. Levin and R. Szeliski. Visual odometry and map correlation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[11] B. Mičušík, D. Martinec, and T. Pajdla. 3D metric reconstruction from uncalibrated omnidirectional images. In *Asian Conf. on Computer Vision (ACCV)*, 2004.

[12] D. Nistér. A minimal solution to the generalised 3-point pose problem. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 560–567, 2004.

[13] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 652–659, 2004.

[14] R. Pless. Using many cameras as one. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 587–593, 2003.

[15] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Intl. J. of Robotics Research*, 21(8):735–758, August 2002.

[16] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.