Online/Realtime Structure and Motion for General Camera Models*

Gerald Schweighofer Siniša Šegvić Axel Pinz
Institute of Electrical Measurement and Measurement Signal Processing
Graz University of Technology
gerald.schweighofer@tugraz.at, ssegvic@zemris.fer.hr, axel.pinz@tugraz.at

Abstract

This paper presents a novel algorithm for online structure and motion estimation. The algorithm works for general camera models and minimizes object space error, it does not rely on gradient-based optimization, and it is provably globally convergent. In comparison to previous work, which reports cubic complexity in the number of frames, our major contribution is a significant reduction of complexity. The new algorithm requires constant time per frame and can thus be used in online applications. Experimental results show high reconstruction accuracy with respect to simulated ground truth data. We also present two applications in artificial marker reconstruction and handheld augmented reality.

1. Introduction

This paper presents a novel *Online Structure and Motion* (SaM) algorithm. Due to the high non-linearity involved, best SaM accuracy is obtained through an iterative refinement post-processing step. In most of the previously presented SaM algorithms, the refinement is formulated as a gradient-based optimization procedure which is also known as bundle adjustment. The main shortcoming of bundle adjustment is that it can get stuck in a local minimum, which tends to happen very often. We propose a novel refinement approach based on the *absolute orientation* algorithm by Horn [6]. The corresponding object space cost function is suitable for use in general camera models, which is very convenient for applications involving multiple cameras.

The paper is organized as follows. We briefly review the main results of our previously published "Structure and Motion" algorithm [9] in section 2. This algorithm is globally convergent and up to 8 times faster than bundle-adjustment, which gives it an edge in time-critical applications. In section 3 we present a significant improvement of this algorithm towards online/realtime applications by reducing the cubic time complexity to a constant one. Section 4.1 evaluates accuracy and speed of the novel algorithm based on simulated ground truth data. We continue with a more realistic application in hand-held Augmented Reality in section 4.2, which describes the hardware of our mobile demonstrator and the implementation of the proposed SaM algorithm including natural landmark detection and tracking. We demonstrate how our approach can improve the performance of a marker-based AR system. Finally, we conclude with a brief discussion and give an outlook on future work in section 5.

*This work was supported by the Austrian Science fund S9103-N13 and the European MC IIF project AViCMaL.

2. Review of globally convergent SaM

The SaM algorithm previously proposed in [9] does not rely on gradient-based optimization, but is nevertheless provably globally convergent. The main components of the algorithm are: (i) object space cost function, (ii) general camera model, (iii) closed form structure estimation, (iv) closed form camera translation estimation, and (v) iterative formulation. The main results of [9] are briefly summarized below.

2.1. Object space cost function

The object space cost function [7] is the distance between the *interpretation line* of a pixel and the reconstructed 3D point. The interpretation line of a measured pixel $\vec{\mathbf{v}}_{ki}$ joins it with the optical center c_k of camera k.

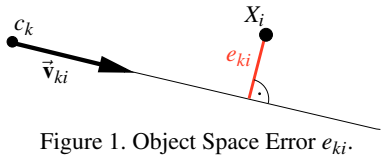


Figure 1. Object Space Error e_{ki}.

This is illustrated in Fig. 1 where X_i represents the estimate

of a landmark 3D position, while e_{ki} stands for the corresponding cost. This error was used in [7, 12] for pose estimation and in [8] for bundle adjustment.

2.2. Object space error for the general camera model

The general camera model (GCM) described in [5] replaces the use of a pixel (which contains the position on the sensor and its intensity) as the atomic element with the *raxel*, which is the union of the pixel and its interpretation line. We represent this *raxel* by a point $\vec{\mathbf{c}}$ (which could be the physical position of the pixel inside the camera, or the center of projection of a perspective camera), and a vector $\vec{\mathbf{v}}$ which defines the ray. This model allows us to use any type of camera geometry (e.g. perspective, catadioptric, panoramic, clusters of cameras, ...).

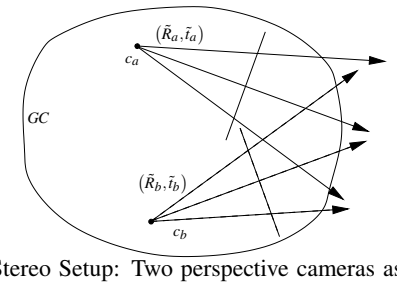


Figure 2. Stereo Setup: Two perspective cameras as one general camera.

In Fig. 2 we see how the model can be used to describe a stereo setup. The two cameras are further addressed as one general camera (GC).

Using the tuple ($\vec{\mathbf{c}}$,$\vec{\mathbf{v}}$) as the measure of a GC, we define the object space cost function for the general camera model as

$$e_{ki} = \left\| (I - V_{ki}) \left(R_k \vec{\mathbf{X}}_i + \vec{\mathbf{t}}_k - \vec{\mathbf{c}}_{ki} \right) \right\|^2, \text{ with } V_{ki} = \frac{\vec{\mathbf{v}}_{ki} \vec{\mathbf{v}}_{ki}^T}{\vec{\mathbf{v}}_{ki}^T \vec{\mathbf{v}}_{ki}}. \quad (1)$$

The matrix V_{ki} is called *line of sight projection matrix*, since it encodes the ray of our measurement in a simple way. R_k and t_k represent the orientation and the translation of a GC.

Assume that a moving GC has measured projections of structure points ($\mathcal{X} = \{X_1, X_2, \ldots, X_{n_i}\}$) at n_k different positions and orientations in space. Then our goal is to use these measurements to recover the scene structure $\mathcal{X}$ and n_k GC poses ($\mathcal{R} = \{R_1, R_2, \ldots, R_{n_k}\}, t = \{t_1, t_2, \ldots, t_{n_k}\}$). We achieve that by minimizing the sum of all costs:

$$\arg\min_{\mathcal{R},t,\mathcal{X}} E(\mathcal{R}, t, \mathcal{X}) = \arg\min_{\mathcal{R},t,\mathcal{X}} \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} e_{ki}(R_k, t_k, X_i). \quad (2)$$

2.3. Closed form Structure estimation

A closed form equation for structure which minimizes (2) can be determined by finding the roots of a partial derivative of (2) with respect to the structure. The resulting structure is given by [9]

$$\tilde{\vec{\mathbf{X}}}_i(\mathcal{R}, t) = \tilde{x}_i(\mathcal{R}) + \sum_{k=1}^{n_k} \tilde{X}_{ki}(\mathcal{R}) \vec{\mathbf{t}}_k, \quad \text{width} \quad (3)$$

$$\tilde{X}_{ki}(\mathcal{R}) = -\left(\sum_{\alpha=1}^{n_k} R_\alpha^T Q_{\alpha i} R_\alpha \right)^{-1} R_k^T Q_{ki}, \quad (4)$$

$$\tilde{x}_i(\mathcal{R}) = \left(\sum_{\alpha=1}^{n_k} R_\alpha^T Q_{\alpha i} R_\alpha \right)^{-1} \sum_{k=1}^{n_k} R_k^T Q_{ki} \vec{\mathbf{c}}_{ki}, \quad (5)$$

$$Q_{ki} = (I - V_{ki})^T (I - V_{ki}). \quad (6)$$

2.4. Closed form estimation of the camera poses

We now substitute the optimal structure (3) into the cost function (2), and derivate the result with respect to the camera translations. By equating this partial derivative to zero, we obtain the following linear equation system:

$$\mathcal{A}\vec{t} = \begin{bmatrix} A_{1,1} & \ldots & A_{1,n_k} \\ \vdots & \ddots & \vdots \\ A_{n_k,1} & \ldots & A_{n_k,n_k} \end{bmatrix} \begin{bmatrix} t_1 \\ \vdots \\ t_{n_k} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{n_k} \end{bmatrix}, \quad (7)$$

with

$$A_{q,k} = \sum_{i=1}^{n_i} \tilde{X}_{qi}^T R_k^T Q_{ki}, \quad A_{q,q} = \sum_{i=1}^{n_i} \tilde{X}_{qi}^T R_k^T Q_{ki} + \sum_{i=1}^{n_i} Q_{qi}, \quad (8)$$

$$b_q = -\sum_{i=1}^{n_i} Q_{qi} (R_q \tilde{x}_i - \vec{\mathbf{c}}_{qi}). \quad (9)$$

This system can be used to solve for camera translations which minimize (2), provided that we know camera rotations.

2.5. The iterative formulation of the SaM algorithm

The last two paragraphs showed that for a given set of camera orientations (R_k) both structure and camera translations can be estimated in closed form. It is proven in [9], that starting from a set of rotations $R_k^{(\lambda)}$ the following iteration step will result in a *globally convergent* algorithm:

$$R_k^{(\lambda+1)} = \arg\min_R \sum_{i=1}^{n_i} \left\| R \vec{\mathbf{X}}_i^{*(\lambda)} + \vec{\mathbf{t}}_k^{*(\lambda)} + \left(-\vec{\mathbf{c}}_{ki} - V_{ki} \vec{\mathbf{q}}_{ki}^{(\lambda)} \right) \right\|^2, \quad \text{subject to } R^T R = I, \quad (10)$$

with

$$\vec{\mathbf{q}}_{ki}^{(\lambda)} = R_k \vec{\mathbf{X}}_i^{*(\lambda)} + \vec{\mathbf{t}}_k^{*(\lambda)} - \vec{\mathbf{c}}_{ki}. \quad (11)$$

Note that $\vec{\mathbf{X}}_i^{*(\lambda)}$ and $\vec{\mathbf{t}}_k^{*(\lambda)}$ stand for optimal estimates for a given iteration (λ), as shown in section 2.3 and 2.4. The absolute orientation problem (10) can be solved using quaternions [6] or singular value decomposition (SVD) [7].

3. Online SaM

Note that the summations in (3)-(5) range over all frames n_k in the sequence and that the summations in (8) and (9) range over all points n_i. This leads to a time complexity of $O(n_k^3 + n_k n_m)$, with n_m the number of measurements (if all points are visible in all images $n_m = n_i n_k$, but typically $n_m \ll n_i n_k$). This shows that the algorithm proposed in [9] is not usable in an online system, because it slows down when more and more frames are added.

In this section we present an extension to the proposed algorithm which gets rid of that limitation. In an online system the frames are added one by one to the system. After a frame is added the cost in (2) is minimized. As more and more frames are added, we observe that the first frames are not affected by the optimization procedure any more. They stay constant. That may happen because (i) they are already well estimated, or (ii) points from these frames are not visible anymore, or (iii) points from these frames are still visible but are not changed while optimizing. A frame will only change its position or orientation if any of the points visible inside that frame gets changed. If that does not happen, the frame will not be changed during optimization.

A well suited assumption to such an online SaM system therefore is that frames, which were added a long time ago are kept constant within the current optimization procedure. To achieve this behaviour we split the set of all frames into two sets: (i) frames which stay constant $F_c := \{1, 2, \ldots, n_k - f_o\}$, and (ii) frames which are optimized $F_o := \{n_k - f_o + 1, \ldots, n_k\}$ ($f_o = |F_o|$).

Using these two sets in the structure estimate (3) gives

$$\tilde{\vec{\mathbf{X}}}_i(\mathcal{R}, t) = \mathcal{M}_i^{-1} \left(\mathcal{RQ}\mathfrak{c}_i + \mathcal{RQ}\mathfrak{t}_i \right), \tag{12}$$

with

$$\mathcal{M}_i = \sum_{k \in F_c} R_k^T Q_{ki} R_k + \sum_{k \in F_o} R_k^T Q_{ki} R_k, \tag{13}$$

$$\mathcal{RQ}\mathfrak{t}_i = \sum_{k \in F_c} R_k^T Q_{ki} \vec{\mathbf{t}}_k + \sum_{k \in F_o} R_k^T Q_{ki} \vec{\mathbf{t}}_k, \tag{14}$$

$$\mathcal{RQ}\mathfrak{c}_i = \sum_{k \in F_c} R_k^T Q_{ki} \vec{\mathbf{c}}_{ki} + \sum_{k \in F_o} R_k^T Q_{ki} \vec{\mathbf{c}}_{ki}. \tag{15}$$

In these equations the summations over the set F_c can be pre-calculated, they need not be evaluated inside the optimization loop.

Rewriting the estimation of the camera translation in (7) is a bit more complicated. To do that we split the translation vector into two parts. These are the translations $t_c := \left[t_1^T, \ldots, t_{n_k - f_o}^T \right]^T$ which are kept constant and the translations $t_o := \left[t_{n_k - f_o + 1}^T, \ldots, t_{n_k}^T \right]$ which we want to optimize. The linear equation system (7) becomes

$$[\mathcal{A}_1 | \mathcal{A}_2] \begin{bmatrix} t_c \\ t_o \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \tag{16}$$

Since t_c stay constant, a solution of the unknown translations t_o is given by a solution to the lower part of the equation system (16)

$$\mathcal{A}_2 t_o = \mathcal{A}_1 t_c + b_2 = \begin{bmatrix} bb_{n_k - f_o + 1}, & \ldots, & bb_{n_k} \end{bmatrix}^T. \tag{17}$$

Using the variables defined in (13)-(15) the entries of $\mathcal{A}_2$ are given by

$$A_{q,k} = -\sum_{i=1}^{n_i} Q_{qi} R_q \mathcal{M}_i^{-1} R_k^T Q_{ki}, \tag{18}$$

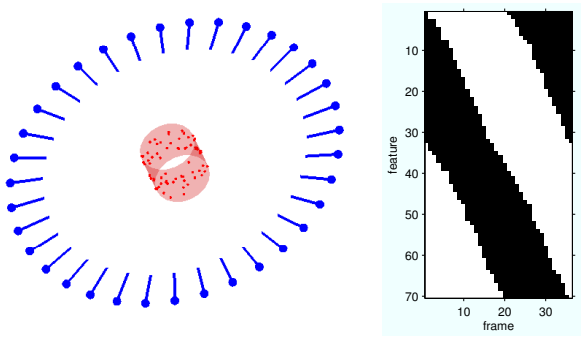
$$A_{q,q} = -\sum_{i=1}^{n_i} Q_{qi} R_q \mathcal{M}_i^{-1} R_q^T Q_{qi} - \sum_{i=1}^{n_i} Q_{qi}, \tag{19}$$

and the entries of bb_q are given by

$$bb_q = \sum_{i=1}^{n_i} Q_{qi} c_i - Q_{qi} R_q \mathcal{M}^{-1} \left(\mathcal{RQ}\mathfrak{c}_i - \sum_{\alpha \in F_c} R_\alpha^T Q_{\alpha i} t_\alpha \right). \tag{20}$$

Using the results above we are able to propose the "Online Structure and Motion for GCM" algorithm:

1. Split the set of all frames into two subsets F_o and F_c.
2. Precalculate the summations over the set F_c in (13)-(15) and (20).
3. Start with an initial set of rotations $R_{k \in F_o}^{(\lambda)}$, $\lambda = 1$. R_{n_k} is initialized with $R_{n_k - 1}$ (the last known frame), or with the result of a pose estimation algorithm using the estimated structure (we used the algorithm proposed in [10], which estimates the pose of a camera using the same cost function as we do, the object space cost). The other rotations are used from results of previous optimizations.
4. Estimate the parameterization of the structure ($\mathcal{M}_i$, $\mathcal{RQ}\mathfrak{c}_i$ and $\mathcal{RQ}\mathfrak{t}_i$) for all points visible in frames F_o using (13)-(15)
5. Estimate the optimal translation vectors $\vec{\mathbf{t}}_{k \in F_o}^{*(\lambda)}(\mathcal{R}^\lambda)$ solving the linear equation system in (17).
6. Estimate an optimal structure $\vec{\mathbf{X}}_i^{*(\lambda)}(\mathcal{R}^\lambda, \vec{t}^{*(\lambda)})$ using (12) for all points visible in frames F_o.
7. Solve the *absolute orientation problem* in (10) for each camera position $k \in F_o$ to get better estimates of the rotations $\mathcal{R}_{k \in F_o}^{(\lambda+1)}$.
8. Set $\lambda = \lambda + 1$ and goto step 4 until convergence or a fixed number of iterations are performed.



(a) (b)

Figure 3. Setup of the simulation. (a) 3D view, (b) visibility map.

Implementation details: The first frame of a sequence is directly added to the set F_c. It is used as a reference frame for the rest of the algorithm. Each new frame n_k, which is the last up to that point in the sequence, is added to the set F_o. If the set F_o grows above f_o ($|F_o| > f_o$) then the oldest frame in F_o is moved to the set F_c. If that happens, then the summations over set F_c ((13)-(15) and (20)) are updated. This update is of constant cost, because only a new summand needs to be added. After that the steps 4 to 8 are performed.

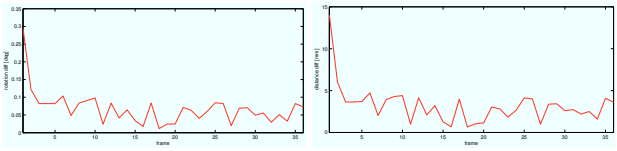
In this implementation the refinement (10) concerns the poses of the last f_o frames, while the poses of the preceding frames are not changed. However, the "fixed" frames F_c are still used to evaluate the structure uncertainty. The object space error (3) reflects the evidence obtained during the entire feature lifetime, which includes the fixed frames. Due to the employed object-space error formulation, the structure update can be expressed by simple equations, which, by careful caching (as described above), can be implemented in constant time.

4. Experiments

Two types of experiments are performed. First, we describe the performance of the proposed algorithm using simulated data. With simulated data we have a setup to compare the results with the ground truth. Second, we explain the behavior of the proposed algorithm in two applications.

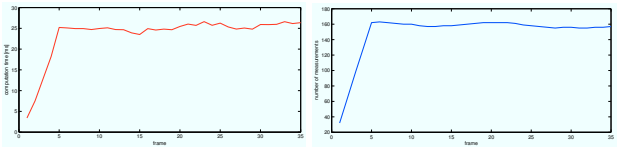
4.1. Simulation

The used setup was motivated by the experiments done in [4]. There a camera was moving around an object. To simulate that behavior we generated 70 random points on the surface of a cylinder with a diameter and a height of one meter. The general camera (we used a stereo rig) is placed at a distance of three meters from the center of the cylinder, and rotated around it to generate 36 frames, one each 10^o.



(a) (b)

Figure 4. Recovered accuracy of the simulation. (a) camera rotation (b) camera position.



(a) (b)

Figure 5. Timings of the proposed algorithm. (a) consumed CPU time per frame, (b) number of measurements used inside the optimization.

In Fig. 3(a) a 3D view of the simulation is shown. The blue points show the camera center (one of the two inside our GC) plus the viewing direction. The calibration of the GC is the same as in section 4.2.

For each camera position the 3D points are projected into the cameras. Here only points are used, which are visible by a given camera (points on the other side of the cylinder are not visible). This simulates the observation of an object. In Fig. 3(b) we see the visibility map. A black point indicates that the corresponding 3D point is visible in the corresponding frame. On average, 32 points are seen in each frame. It was not stated in section 3 that the proposed algorithm is usable in such a scenario, in which not all points are visible in all frames. But it is easy to verify that setting $Q_{qi} = 0_{3x3}$ if a 3D point i is not visible in the frame q will turn on that possibility. After projecting the points to the images, Gaussian noise with $\sigma = 0.5$ pixel is added to the measurements. The estimated measurements are fed into the proposed algorithm frame by frame. The number of frames we optimized was set to $f_o = 5$. The algorithm was stopped after performing 20 iterations.

In Fig. 4 we see the reconstruction results of our algorithm. In Fig. 4(a) the difference of the estimated rotation w.r.t. ground truth for all frames is plotted. Fig. 4(b) shows the difference of the estimated position w.r.t. ground truth of all frames. We see that the accuracy is starting with 0.35^o for the first frame and decreasing. After a few frames ($f > 3$) enough information is available and the accuracy stabilizes at a level of 0.1 degrees for the rotation and about 5 mm for the camera position. The comparison of the reconstruction with the ground truth gives a mean error of 0.319 mm.

In Fig. 5(a) we show the consumed amount of CPU time for each frame. After the first few frames are added the required CPU time stays nearly constant at about 25 ms. To

compare this to the number of measurements (which are not constant over all frames) we show in Fig. 5(b) the number of measurements in the set F_o. We see that there is a direct relation between both graphs. This leads to 7.9μs of CPU time per measurement and iteration inside the optimization procedure.

4.2. Applications

The proposed SaM algorithm has been implemented on a mobile demonstrator platform (shown in Fig. 7(a)) which consists of a tablet PC and a calibrated stereo rig. The mobile demonstrator is equipped with a Pentium M Processor running at 1.2GHz, and an $8.4''$ SVGA TFT LCD Touchscreen Display. Its dimensions are $227 \times 170 \times 22mm$. The stereo setup has two UI-1220-C cameras providing 752×480 images at a frame rate up to 60Hz, featuring external trigger, global shutter and USB 2.0. The cameras are calibrated to conform to the general camera model [13]. To be able to use the proposed SaM algorithm we need feature correspondences. For that, in each image the features projected from natural landmarks are detected using the interest operator of Tomasi and Kanade [11]. To improve the execution speed, the detected features are first associated with their positions in the last frame by simple correlation. Consequently, the position is more accurately estimated by aligning the feature with its reference appearance acquired in the image where the feature was first detected. We used the algorithm by Zinßer et al. [14], which combines the inverse compositional gradient descent affine alignment [2] and illumination compensation.
The natural features are detected and tracked in both images. After that, features which correspond to the same $3D$ point are linked together. This is done using epipolar geometry, which is known from the calibration [13], and correlation. The image coordinates (x and y) of the tracked features are normalized to obtain $(\vec{\mathbf{c}},\vec{\mathbf{v}})$, again by using calibration data.
For the two applications below, the pose of the general camera at the initialization stage is recovered [10] from an artificial landmark (ARToolKit Plus [1]) which is detected by a dedicated procedure. After the initialization, the natural landmarks are fed into the SaM algorithm. The recovered pose corresponding to the last frame is finally used in the AR front end.

Bridging Marker-less Environments in AR

This scenario happens very often in marker-based AR systems. As long as a marker is seen by a camera, the pose can be recovered and so the artificial objects can be visualized. If no marker is visible, no pose can be computed.

Our system uses natural landmarks to overcome this problem. In Fig. 6 we see images of a sequence. In the beginning Fig. 6(a)-(b) the marker is visible, and its information is used to compute a pose, which is used together with the tracked natural landmarks to estimate the 3D reconstruction and to generate the augmentation (a virtual plant). When the target disappears Fig. 6(c)-(d) only natural landmarks are used to estimate a pose. In Fig. 6(e)-(g) we see that the plant is correctly visualized. Finally, the marker reappears in Fig. 6(h).

Automatic Generation of Scene Description for AR

The second scenario, which we address in this paper is the automatic generation of a scene description. We start by attaching artificial markers to the walls of our lab. The goal of the system is the automatic generation of the orientation and the position of the markers, so that such a map can be used with an AR system like ARToolKit Plus [1]. To achieve this, we walked with our hand-held mobile demonstrator through our lab. The first marker which is seen by our system is selected to be the reference marker. Its position is by default set to the origin of the scene coordinate system. After that the artificial markers (their four corners) and tracked natural landmarks are employed in the same way. They are fed into the SaM system. Further, if a new marker is seen, its position and orientation is calculated from the four corners estimates. After all markers have been observed by the system, a full scene description can be generated. As an optional post-processing step, an offline optimization can be run to improve the final precision. Fig. 7(b) shows a wide-angle view of the scene (taken by a consumer point-and-shoot camera) and Fig. 7(c) presents the obtained reconstruction of the markers.

5. Conclusion

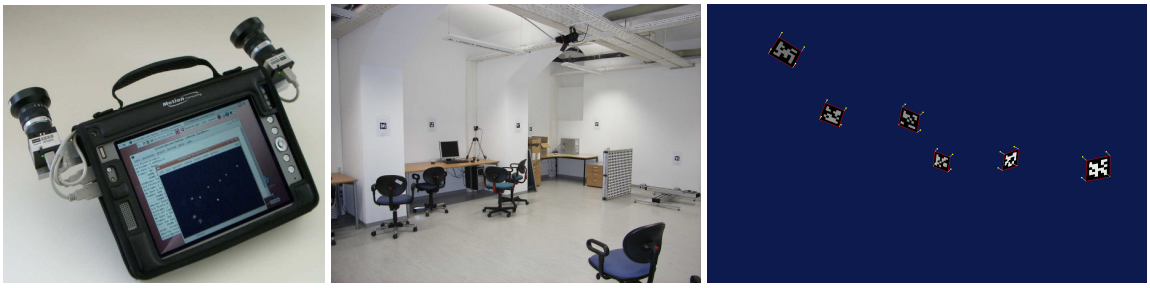
We presented a novel Online Structure and Motion algorithm which is based on the Object Space Error for General Camera Models. The proposed algorithm was implemented and tested with simulated data to obtain information about the accuracy and the timing behavior. A mobile demonstrator was developed to show two relevant applications of online SaM for Augmented Reality. First, augmentation is not only possible in marker-based AR but can be extended to cases where no marker is in the field of view of the cameras. Second, we showed that such a system can be used to automatically generate a 3D reconstruction of the artificial landmarks in the scene.

Future work includes experimental validation of our novel algorithm in navigation (arbitrary motions of the General Camera in a natural scene without artificial markers) and quantitative comparison with [3, 4] in terms of reconstruction accuracy as well as computation time.

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6. AR without a Target. The artificial plant is augmented even if no marker is visible (c)-(g).



(a) (b) (c)

Figure 7. (a) The mobile demonstrator. (a) Wide-angle view of our lab. (b) 3D reconstruction of the artificial markers.

Acknowledgements

We would like to thank Timo Zinßer for providing us with an implementation of [14].

References

[1] ARToolKit Plus. http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php.

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. In *International Journal of Computer Vision*, volume 56, pages 221–255, 2004.

[3] A. J. Davison, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. PAMI*, 29:1052 – 1067, 2007.

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

[5] M. D. Grossberg and S. K. Nayar. A general imaging model and a method for finding its parameters. In *ICCV*, pages 1100–1105, 2001.

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

[7] C. Lu, G. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Trans. PAMI*, 22(6):610–622, 2000.

[8] S. Ramalingam, S. K. Lodha, and P. Sturm. A generic structure-from-motion framework. *Proc. Computer Vision and Image Understanding*, 2006.

[9] G. Schweighofer and A. Pinz. Fast and globally convergent structure and motion estimation for general camera models. In *17th BMVC*, pages 147 – 156, 2006.

[10] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Trans. PAMI*, 28(12):2024 – 2030, 2006.

[11] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[12] P. Wunsch and G. Hirzinger. Registration of cad-models to images by iterative inverse perspective matching. *Proceedings of the 13th ICPR*, 1:78–83, 1996.

[13] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientation. In *Proceedings IEEE ICCV*, pages 666–673, Corfu, Greece, 1999.

[14] T. Zinßer, C. Gräßl, and H. Niemann. High-speed feature point tracking. In *Workshop on Vision Modeling and Visualization*, pages 49–56, Erlangen, 2005.