ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume I-3, 2012
XXII ISPRS Congress, 25 August – 01 September 2012, Melbourne, Australia

REAL-TIME CAMERA GUIDANCE FOR 3D SCENE RECONSTRUCTION

Falko Schindler and Wolfgang Förstner

Department of Photogrammetry
University of Bonn
Nussallee 15, 53119 Bonn, Germany
falko.schindler@uni-bonn.de, wf@ipb.uni-bonn.de
http://www.ipb.uni-bonn.de/[schindler,foerstner]

Commission III/1

KEY WORDS: Three-dimensional Reconstruction, Bundle Adjustment, Camera Orientation, Real-time Planning

ABSTRACT:

We propose a framework for operator guidance during the image acquisition process for reliable multi-view stereo reconstruction. Goal is to achieve full coverage of the object and sufficient overlap.
Multi-view stereo is a commonly used method to reconstruct both camera trajectory and 3D object shape. After determining an initial solution, a globally optimal reconstruction is usually obtained by executing a bundle adjustment involving all images. Acquiring suitable images, however, still requires an experienced operator to ensure accuracy and completeness of the final solution.
We propose an interactive framework for guiding unexperienced users or possibly an autonomous robot. Using approximate camera orientations and object points we estimate point uncertainties within a sliding bundle adjustment and suggest appropriate camera movements. A visual feedback system communicates the decisions to the user in an intuitive way.
We demonstrate the suitability of our system with a virtual image acquisition simulation as well as in real-world scenarios. We show that when following the camera movements suggested by our system, the proposed framework is able to generate good approximate values for the bundle adjustment, leading to accurate results compared to ground truth after few iterations. Possible applications are non-professional 3D acquisition systems on low-cost platforms like mobile phones, autonomously navigating robots as well as online flight planning of unmanned aerial vehicles.

1 INTRODUCTION

Within the last few years, the field of photogrammetry and computer vision highly benefited from cheap and fast hardware in personal computers, cameras and even mobile phones. Capturing devices often play the role of smart sensors, as they control acquisition parameters such as focus, aperture or shutter speed, perform image restoration tasks, or support taking panoramas. Automatic visual reconstruction of objects and scenes is gaining interest for industrial applications and amateur photographers. Therefore a camera system for 3D reconstruction needs to be applicable especially for unexperienced users. We propose a framework for operator guidance during the image acquisition process with the goal to achieve full coverage of the object and sufficient overlap for reliable multi-view stereo reconstruction.

Camera poses for multi-view stereo need to fulfill some requirements to allow for a complete and accurate reconstruction:

- Scene points have to be captured from different viewing directions in order to achieve high enough accuracy and reliability.
- Views need to be linked with sufficiently many common points in order to allow for a joint estimation of all parameters.

Those requirements often are not fulfilled in case the images are captured by an unexperienced operator. Moreover, there are no simple and applicable rules for ensuring overlap and accuracy.

We propose an automatic, interactive camera system that guides the operator through the scene. Based on video image stream the system automatically detects key frames and captures high-resolution images to be used for final bundle adjustment. To do so, it reconstructs both camera trajectory and scene structure in real-time in order to make decisions about how and where to move next. Those decisions are passed to the operator in form of visual advices.

In contrast to many other works on localization and mapping we exploit the live interaction between the user and the camera system. By guiding the operator interactively the camera system can suggest viewing poses, position and orientation, for reconstructions of a required quality. Furthermore the system immediately yields an approximate solution for a global bundle adjustment including all cameras.

2 RELATED WORK

Multi-view stereo reconstruction is one of the main expertises in photogrammetry and computer vision. In the last few years this technique became available for large-scale photo collections (Agarwal et al., 2009; Frahm et al., 2010). Much effort has been put into selecting suitable images of the object of interest and preparing an approximate solution for the final bundle adjustment. For the ROME dataset used by Frahm et al. (2010) about 22 hours of computing time are needed for clustering, geometric verification and building the iconic scene graph, before performing a dense reconstruction within 2 hours.

In these scenarios the images were available on the internet, of course with no view on the final reconstruction whatsoever, partly leading to weak geometries or incomplete reconstructions. When trying to achieve full coverage and high enough accuracy, the question arises: Can we make use of the fact that an operator

Figure 1: Microsoft PhotoSynth application for mobile devices as an example for an intuitive photogrammetric application for unexperienced users. The operator is asked to turn the viewing direction (green dot) out of the previously captured area (highlighted polygon). When the dot reaches the dashed line, a new image is taken automatically, preserving enough overlap of the new image (green rectangle) with the others (white outline) for the post-processing image stitching. The user may take additional photos which then are merged to the automatically taken ones.

is taking images at this instant, one image after the other, while moving through the scene? This at the same time would allow the reconstruction framework incrementally build a coarse reconstruction and derive guidance instructions for the operator from a first, coarse reconstruction.

Therefore the reconstruction framework needs to track the moving camera and reconstruct parts of the scene in real-time, as proposed recently with, e. g., Parallel Tracking and Mapping (Klein and Murray, 2007) and Dense Tracking and Mapping (Newcombe et al., 2011). Those systems are impressively fast, robust and accurate. They are, however, designed for small desktop environments and not necessarily comfortable and useful for outdoor and large-scale scenarios.

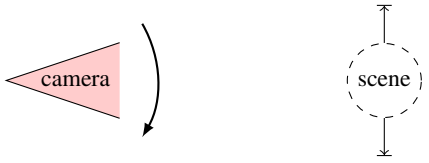
Since we possibly want to run our system on low-powered integrated camera devices, we restrict to a less complex approach for real-time tracking. In case of loosing orientation we always can interactively ask the operator to go back to the last known position. Since we explicitly make use of the operator's attention, this is no draw-back.

In order to make such a system accessible to even unexperienced operators, we chose to design the system as simple and intuitive as possible. We got inspiration from the Microsoft PhotoSynth[1] application for mobile devices. This software enables the user to take panorama images by automatically triggering the camera, while he is turning around himself. The game-like approach transfers the difficult problem of taking suitable images in different directions with sufficient overlap to a simple 2D "move the point out of a polygon"-task (Fig. 1). Although our objective is a bit more complex, we formulate two game-like tasks of similar simplicity.

3 THEORY

Our proposed camera system consists of a camera, being able to capture high-resolution images, a low-resolution video stream,

[1] `http://itunes.apple.com/de/app/photosynth/id430065256`



(a) In explore mode the operator is asked to turn around his own position in order to explore new areas of the scene.

(b) In refine mode the operator is asked to move around the currently observed area of the scene in order to refine the scene point locations.

Figure 2: Schematic overview of the two tasks the operator needs to fulfill by turns. The camera (left) needs to move differently round the objects (right).

and a display for showing the live camera image and visual hints. The video stream is used to track KLT features (Lucas and Kanade, 1981) and to determine an approximate solution for the current camera pose and observed scene points. After *significant movement* a high-resolution image is taken for the offline bundle adjustment. The significance of a movement is determined by checking coverage and accuracy.

3.1 Two tasks: Explore and refine scene points

The operator needs to fulfill two tasks: First, the camera has to capture new scene points without loosing sight of previously located points: The operator needs to *explore* the scene. Second, each new point has to be viewed from significantly different viewing directions in order to get an accurate location: The operator needs to *refine* those points.

A significant movement is defined differently, depending on which of these two tasks the system is currently focusing on:

Explore mode When all previously observed scene points are sufficiently accurately determined, the systems asks the operator to capture new points by turning around his own position, e. g. around the vertical axis, as shown in Fig. 2(a). As soon as sufficiently many new points appeared within the field of view or a minimum number of previously located points is reached, a new key frame is triggered.

Refine mode When some new scene points are not located accurately enough, the system asks the operator to refine those points by moving around them, as shown in Fig. 2(b). As soon as the angle between previous and current viewing direction exceeds a certain threshold, e. g. 5 degrees, a new key frame is triggered.

A scene point is sufficiently accurately observed when all eigenvalues of its covariance matrix are below some threshold. If the scale of the observed scene is known, e. g. via user input, this threshold is metric and can be set to the desired reconstruction accuracy. Otherwise the accuracy criteria can also be defined depending on the isotropy of the point's covariance Beder and Steffen (2006).

After exploring new points, the system always switches into refine mode. After refine mode, however, the system may remain in that mode if changing the viewing direction by some degrees did not sufficiently increased the accuracy of the point locations.

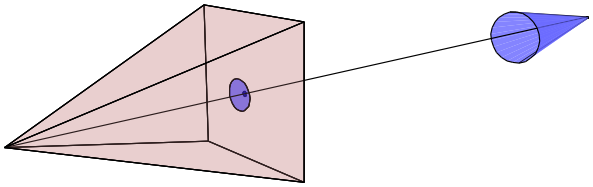


Figure 3: Visualization in refine mode. A virtual cone is generated starting at one selected scene point (right) and pointing with its circular ground surface towards the camera projection center of the previous key frame. The opening angle of the cone is defined by the angular threshold for the required difference of consecutive viewing directions in refine mode, e. g. 5 degrees. The length of the cone is not essential. It is set to some fraction of the scene point's distance to the camera. Both the selected scene point (black dot) and the cone's ground circle (blue circular patch) are projected into the current camera view. As the operator approaches the required change in viewing direction, the dot is moving out of the circle which is an intuitive feedback for solving the refinement task.

3.2 Key frame detection

After defining the tasks the operator needs to complete, we are now looking into the task-specific key frame detection in more detail.

Significant baseline in refine mode In refine mode the operator is asked to move around the scene in order to change the viewing direction significantly. This is almost equivalent to achieving a significant baseline between consecutive key frames, but is independent of the unknown scale and avoids movements straight towards the scene points which does not improve the configuration.

To give the operator an intuitive feedback whether he approaches or retreats the next key frame, we visualize the threshold of 5 degrees change in viewing direction in form of an ellipse around an automatically selected KLT point. The point is one that needs refinement and is closest to the image center. The ellipse is generated by projecting a cone into the image, as depicted in Fig. 3.

When moving around the object, the operator tends to escape the virtual cone. Within the projected image the dot is escaping the projected circle which is the game-like task the operator needs to solve. This is sufficient for control, as moving towards the object does not improve the situation; the visual feedback suggests not to continue, since the dot remains fixed within the circle. Going back to the previous position also does not improve the camera configuration; in the camera image the dot approaches the center of the circle, suggesting to move into a different direction. Rotating the camera moves both dot and circle, but does not lead to solving the refinement task, since rotation does not improve the scene point localization.

Enough new points in explore mode In explore mode the operator is asked to turn around himself in order to capture new areas of the scene, but still preserving overlap with previous frames.

We display the convex hull of all old reconstructed KLT points and highlight the newly detected points. By turning towards new areas of the scene some old points get lost and are replaced with new ones. The exploration task is fulfilled as soon as a certain amount of new points is captured outside of the convex hull of the old points. The displayed polygon of the convex hull is updated in real-time.

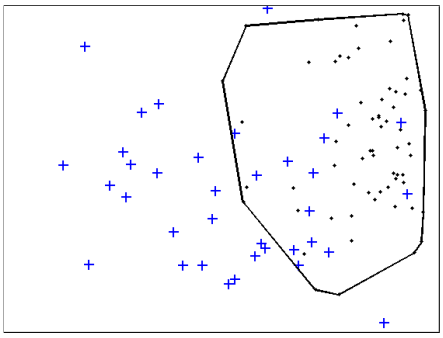


Figure 4: Visualization in explore mode. The convex hull (black polygon) of all points observed in previous frames (black dots) is displayed. New points being observed in the current frame are highlighted with a blue "+". The exploration task is to turn the camera and catch new points outside the old bounding box.

3.3 Approximate solution

In order to be able to automatically detect key frames based on the above-mentioned criteria we need to obtain real-time solutions for camera poses and scene point locations. While the second camera is oriented relatively w. r. t. the first one, the next cameras are oriented via spatial resectioning. Thus only the scale of the first baseline remains unknown and propagates into the scene points and the remaining cameras.

Relative camera orientation The relative orientation of the second camera is determined according to (Nister, 2004). Therefore the essential matrix is computed from multiple minimal samples of 5 KLT tracks. We choose the solution that minimizes the sum over the Huber-norm of the re-projection errors, see (Torr and Zisserman, 2000).

Since KLT features are tracked quite robustly within video sequences, we can assume a high inlier rate of, e. g., 50 % and only need 218 random samples with a RANSAC approach (Fischler and Bolles, 1981) for finding a correct solution with a certainty of 99.9 %.

Absolute camera orientation The absolute camera orientation is determined via spatial resection with at least 3 known scene points, again minimizing the Huber norm. We use Grunert's direct solution (Grunert, 1841) as summarized in (Haralick et al., 1994).

In contrast to determining the relative orientation with a minimal set of 5 homologous points, we need only 3 correctly tracked points for computing the spatial resection of the current camera. Under same assumptions as before we now need 52 random samples for finding a correct solution with a certainty of 99.9 %.

Scene points via forward intersection Scene point coordinates are determined via forward intersection from at least two cameras. Each image ray $\mathbf{m}_{c,i}$ from camera c to scene point i has to be parallel to the projection of the homogeneous scene point $\mathbf{X}_i$ using the normalized projection matrix $\mathsf{P}_c = R_c^\mathsf{T}[I_3 \mid -\mathbf{Z}_c]$ of camera c specified by rotation matrix R_c and projection center $\mathbf{Z}_c$:

$$\mathbf{0} = \mathsf{S}(\mathbf{m}_{c,i})\mathsf{P}_c\mathbf{X}_i. \quad (1)$$

The parallelism is enforced by restricting the vector product – formulated using the skew symmetric matrix $S : S(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$ – to be zero. Combining equations for each of the C cameras observing one point $\mathbf{X}_i$ yields the point $\mathbf{X}_i$ as the right singular vector to the smallest singular value of the $(3C \times 4)$-matrix $[\mathsf{S}(\mathbf{m}_{c,i})\mathsf{P}_c]$, see (McGlone, 2004, eq. (11.103)).

This way we possibly obtain points at infinity or even behind the camera. By checking the angle between image ray $\mathbf{m}_{c,i}$ and the connection between camera projection center and scene point $\mathbf{X}_i$ we identify and remove points behind on of the cameras. Since points at infinite distance or with small intersection angles can be handled by the following bundle adjustment, we do not have to remove them.

3.4 Bundle adjustment

After detecting a new key frame, we run a bundle adjustment with all cameras and scene points. Optionally one can restrict to the last k key frames, i. e. doing a sliding bundle adjustment, leading to constant computing time and memory requirements.

We apply a novel bundle adjustment (Schneider et al., 2011), allowing for various types of cameras, even multi-camera systems and – most importantly – points at infinity. The latter is useful for outdoor applications, when very distant points are observed over long periods of time, since those points yield a very stable estimate for the camera rotation parameters.

3.5 Algorithm

Let us summarize the overall algorithm for interactively tracking the moving camera, reconstructing scene points and guiding the operator. A schematic description is shown in Algorithm 1.

Initially we start with a first key frame and define the corresponding camera pose to be at the origin of the global coordinate system. The scene point cloud is empty.

Until a new key frame is detected, the program remains repeating three steps:

1. Image points are observed. Therefore new coordinates for tracked KLT features are read from the video camera module. Depending on the current mode, lost KLT tracks are allowed or disallowed to be replaced.
2. The camera is oriented approximately. For the second key frame the orientation is computed relatively to the first camera. Later we compute an absolute orientation w. r. t. previously forward intersected scene points.
3. The program decides, whether a new key frame is detected. In refine mode the baseline, or the change in viewing direction respectively, is checked. In explore mode we count the number of new KLT points outside the convex hull of old KLT points.

After detecting a new key frame, we forward intersect new scene points and run a bundle adjustment with all observations made so far. We repeat waiting for new key frames and updating the reconstructed scene, until the operator manually interrupts this process.

3.6 Implementation details

We implemented the proposed framework for MathWorks[2] Matlab. KLT points, however, are detected and tracked in a separate thread, maintained from a C++ routine via the MEX-interface. This allows to run the feature tracker with about 30 Hz, while the processing of one frame within the Matlab routines can take more time, e. g. 2 to 5 Hz, without delaying the real-time tracking.

[2] http://www.mathworks.com/

```
// initialize
set first camera to origin;
allocate empty scene point cloud;

// infinite loop
repeat
    // wait for new key frame
    repeat
        // observe image points
        if refine mode then
            track KLT features, do not replace lost features;
        else if explore mode then
            track KLT features, replace lost features;

        // orient current camera
        if second key frame then
            compute relative orientation to first camera;
        else
            compute absolute orientation;

        // detect key frame
        if refine mode then
            if significant baseline then key frame detected;
        else if explore mode then
            if enough new points then key frame detected;
    until new key frame detected ;

    // update scene
    forward intersect new points;
    run bundle adjustment;
until user interruption ;
```

Algorithm 1: Overview of the proposed camera framework. See Section 3.5 for a detailed description.

4 EXPERIMENTS

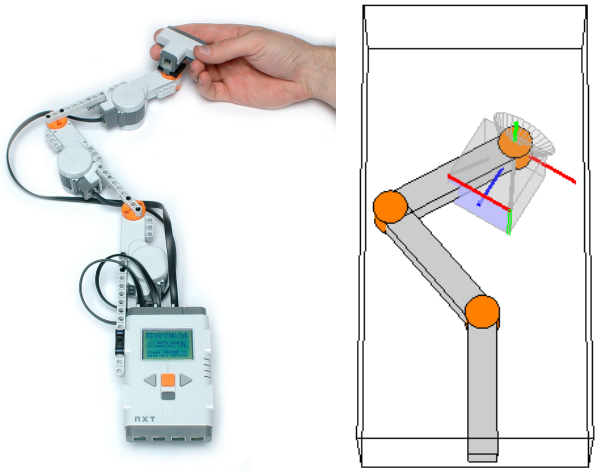
We run two kinds of experiments. First we simulate tracking and reconstruction with a synthetic camera, enabling quantitative comparisons to ground truth orientation parameters. Second we connect a hand-held camera to our tracking framework and investigate our system with real images.

4.1 Simulating camera motions with a passive robot arm

For testing and evaluation purposes we engineered a novel, interactive system for simulating controlled camera motions. We built a simple robotic arm with three rotary sensors, i. e. three degrees of freedom in the 2D plane (Fig. 5). The sensors are connected to a control unit and accessible from our Matlab implementation. For building the robotic arm we used the LEGO Mindstorms NXT[3] system and a Matlab interface from the Institute of Computer Vision[4] at RWTH Aachen University.

From these three sensor values we compute ground-truth projection matrices, generate image points of *virtual* scene points and disturb these perfect observations with errors of known stochastic properties. Contrary to pre-computing a fixed trajectory with camera poses at each instance of time, we can manually steer this virtual camera in real-time and interactively react on visual advices the interactive camera framework is giving us. This way we

[3] http://mindstorms.lego.com/
[4] http://www.lfb.rwth-aachen.de/



(a) Photo of the robotic arm (b) Rendering in Matlab

Figure 5: Photo (left) and rendering (right) of the robotic arm. Rotation angles of the three orange sensors are transferred via the control unit at the bottom to the Matlab implementation of our tracking and reconstruction framework. The pose of the virtual camera at the end of the arm is computed from the three angles and can be used as ground truth camera orientation. Image points are then generated by projecting a virtual scene point cloud into the virtual camera and adding Gaussian noise.

can develop and evaluate the camera tracking modules and the high-level control framework without the need for perfect feature tracking results.

4.2 Real video camera

After successfully testing the camera framework with a simulated camera, we ran the software for a video stream coming from a real hand-held camera (Fig. 6).

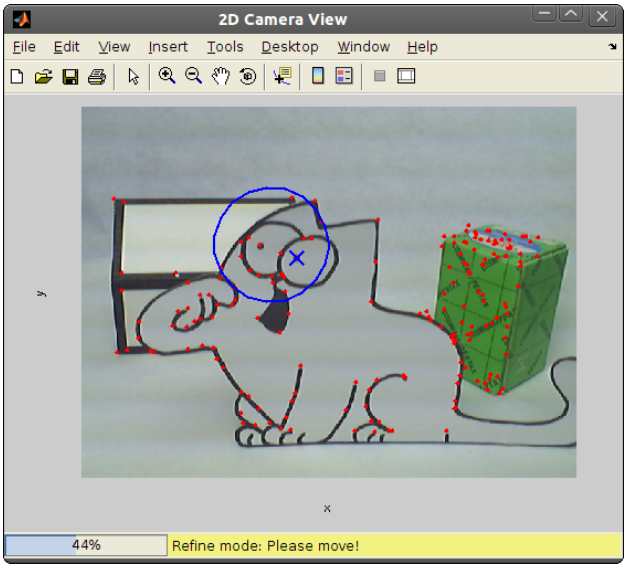
4.3 Quantitative evaluation

The following evaluation is based on the approximate solution or – in case of the simulated camera – ground truth data. Since the absolute position, rotation and scale of relatively oriented cameras and scene points is arbitrary, we transformed both scenes into one common coordinate frame using a spatial similarity transform using all common scene points.

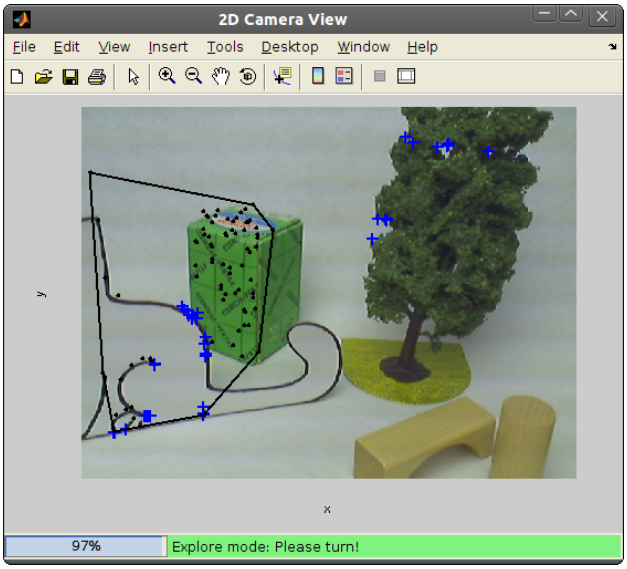
Finally we computed three characteristic measures for the evaluation: (1) mean angles between the camera axes, (2) mean distance of camera projection centers and (3) mean distance of scene point locations. Angles are converted to degrees for easier interpretation. In order to obtain distances independent of the unknown scale factor, we converted distances to percent relative to the coordinate range of all camera projection centers.

For both synthetic and real cameras we assume an observation accuracy of 0.021 degrees. This corresponds to an image point accuracy of 0.3 pixels at 820 pixels focal length. In the following we evaluate two synthetic and one real data set. All of them are summarized in Table 1. While SYNTH1 was created with very slow and careful movements, SYNTH2 is characterized by a fast and hectically moving camera, often leading to very few overlapping points between key frames.

Quality of approximate solution with synthetic camera Comparing the reconstructed scenes before and after performing the bundle adjustments indicates the quality of the algorithms for



(a) Camera view in refine mode. Red dots show inaccurate scene points. The blue cross highlights the points to be focused on right now (pupil of the cat's right eye). This point needs to be brought out of the blue circle, depicting the viewing cone of the previous frame, in order to achieve a significant change in the viewing direction.



(b) Camera view in explore mode. Black dots show old scene points, i. e. observations from previous frames. A black polygon visualizes their convex hull. Newly observed points are marked with a blue "+". While the operator is turning, points on the cat's head are lost and replaced with observations on their back but also an other objects like the tree. The points on the back do not count, since they lie within the convex hull of the old points. But as soon as there are enough new points on the tree, the task is fulfilled.

Figure 6: The proposed camera framework on a real example in both possible modes. The camera image is overlaid with task-specific annotations. The status bar displays the task for the operator and the progress he is doing.

generating approximate values. For SYNTH1 we observe deviations for projection centers and scene points of less than 1 %. Rotation angles deviate in the order of the observation accuracy of 0.021 degrees. Thus the approximate reconstruction is very close to the best estimate, allowing fast conversion of the bundle adjustment within few iterations. See Table 2 for more details on this experiment.

	# cameras	# scene points	# rays	motion
SYNTH1	24	479	2005	careful
SYNTH2	24	473	1921	rough
REAL	14	452	2400	

Table 1: Overview of the three evaluated data sets.

	minimum	median	maximum
camera pitch	0.006°	0.037°	0.232°
camera yaw	0.004°	0.019°	0.338°
camera roll	0.010°	0.041°	0.334°
camera centers	0.095 %	0.382 %	6.335 %
scene points	0.104 %	0.284 %	2.654 %

Table 2: Deviations of three estimated camera orientation angles, camera projection centers and scene point locations of the SYNTH1 data set w. r. t. the approximate solution. Locations are in percent compared to the coordinate range of all cameras.

Quality of final estimation with synthetic camera Our synthetic camera setup allows to compare the final estimation with ground truth data. For SYNTH1 we observe larger deviations of the estimation to the ground truth than to the approximate solution, mostly due to quasi systematic errors caused by the local definition of the coordinate system. After capturing enough key frames, however, we obtain a stable configuration with angles better than 0.1 degrees and positions better than 1 %.

Quality of final estimation with real camera For the data set REAL, captured with a real hand-held camera, we obtain approximate values closer than 1 degree and 10 % to the final estimation. The comparably large deviations in the positions of the cameras can be explained by the small bases between consecutive key frames. During refine mode there are some more extreme deviations up to 5 degrees for angles and 50 % for locations. In explore mode, however, the system recovers again to a more stable reconstruction. See Table 3 for more detailed numbers on this experiment.

	minimum	median	maximum
camera pitch	0.112°	0.810°	4.794°
camera yaw	0.109°	0.829°	3.827°
camera roll	0.182°	1.021°	6.003°
camera centers	0.834 %	8.390 %	54.931 %
scene points	0.251 %	2.563 %	21.379 %

Table 3: Deviations of three estimated camera orientation angles, camera projection centers and scene point locations of the REAL data set w. r. t. the approximate solution. Locations are in percent compared to the coordinate range of all cameras.

Influence of operator behaviour For our synthetic camera setup we did another test SYNTH2 with faster, more careless movements and smaller overlaps between key frames. The approximation is up to 4 times worse than for SYNTH1. The final estimation, however, yields almost same accuracies.

5 CONCLUSION

We proposed a camera framework for capturing a 3D scene that is able to track the camera in real-time, extracts key frames automatically and gives intuitive tasks to the operator in order to improve the quality of a post-processing multi-view reconstruction. A Matlab implementation is running at 2 to 5 Hz, without slowing down the C++ KLT tracker running at 30 Hz in a separate thread.

For testing and evaluating the system we developed a novel robotic interface to manually steer a virtual camera based on a robotic arm with three rotary sensors yielding ground truth camera poses in real-time.

In future work we will try to improve the KLT point selection in order to avoid small clusters with many points and large areas without any. Currently the program stops when the connection to previous frames gets lost, i. e. when not enough homologous points are observed. Furthermore we plan to include inertial sensors for improving robustness and accuracy of the approximate solution.

References

Agarwal, S., Snavely, N., Simon, I., Seitz, S. and Szeliski, R., 2009. Building rome in a day. In: International Conference on Computer Vision.

Beder, C. and Steffen, R., 2006. Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence. In: DAGM Symposium.

Fischler, M. and Bolles, R., 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications Of The Acm 24, pp. 381–395.

Frahm, J., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y., Dunn, E., Clipp, B., Lazebnik, S. and Pollefeys, M., 2010. Building rome on a cloudless day. In: European Conference on Computer Vision.

Grunert, J. A., 1841. Das Pothenotische Problem in erweiterter Gestalt nebst über seine Anwendungen in der Geodäsie. Grunert's Archiv für Mathematik und Physik 1, pp. 238–248.

Haralick, B. M., Lee, C.-N., Ottenberg, K. and Nölle, M., 1994. Review and analysis of solutions of the three point perspective pose estimation problem. International Journal Of Computer Vision 13, pp. 331–356.

Klein, G. and Murray, D., 2007. Parallel tracking and mapping for small AR workspaces. In: International Symposium on Mixed and Augmented Reality.

Lucas, B. D. and Kanade, T., 1981. An iterative image registration technique with an application to stereo vision. In: International Conference on Artificial Intelligence.

McGlone, C. J. (ed.), 2004. Manual of Photogrammetry. American Society of Photogrammetry and Remote Sensing.

Newcombe, R., Lovegrove, S. and Davison, A., 2011. DTAM: Dense tracking and mapping in real-time. In: International Conference on Computer Vision.

Nister, D., 2004. An efficient solution to the five-point relative pose problem. Ieee Transactions On Pattern Analysis and Machine Intelligence 26, pp. 756–770.

Schneider, J., Schindler, F. and Förstner, W., 2011. Bündelausgleichung für Multikamerasysteme. In: DGPF Conference.

Torr, P. H. S. and Zisserman, A., 2000. MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding 78, pp. 138–156.