An Analysis-by-Synthesis Camera Tracking Approach Based on Free-Form Surfaces

Kevin Koeser, Bogumil Bartczak, and Reinhard Koch

Institut für Informatik, Christian-Albrechts-Universität Kiel
D-24098 Kiel, Germany
{koeser,bartczak,rk}@mip.informatik.uni-kiel.de

Abstract. We propose a model-based camera pose estimation approach, which makes use of GPU-assisted analysis-by-synthesis methods on a very wide field of view (e.g. fish-eye) camera. After an initial registration, the synthesis part of the tracking is performed on graphics hardware, which simulates internal and external parameters of the camera, this way minimizing lens and perspective differences between a model view and a real camera image. We show how such a model is automatically created from a scene and analyze the sensitivity of the tracking to the model accuracy, in particular the case when we represent free-form surfaces by planar patches. We also examine accuracy and show on synthetic and on real data that the system does not suffer from drift accumulation. The wide field of view of the camera and the subdivision of our reference model into many textured free-form surfaces make the system robust against moving persons and other occlusions within the environment and provide a camera pose estimate in a fixed and known coordinate system.

1 Introduction and Previous Work

Camera tracking is nowadays used in many applications, e.g. robotics, visual navigation or augmented reality [16,8]. It can suffer from poor localization of visual features, ill-posed estimation (aperture problem, too small field of view), drifting references and occluded or moving scene content. To overcome these issues we propose a fish-eye camera as a visual pose sensor, which captures the surrounding scene in an offline phase and can then be used in an online phase for real-time tracking. Fish-eye cameras have the advantage that they have a very wide field of view compared to a standard perspective camera. Therefore they always "see" large parts of the static scene even if objects or persons move and occlude parts of the background and when the camera rotates. Furthermore, pose estimation is better conditioned than for perspective cameras [9,17,18]. In our approach, the scene in which the camera moves does not need to be set up with expensive calibrated markers as in [3] and can therefore be any location which provides textured surfaces for tracking, e.g. outdoor in front of a building.

During the last years several online camera tracking systems have been proposed: Commercially available systems (e.g. [3]) need special calibrated markers, fast structure-from-motion [1] is prone to drift on long sequences due to a missing absolute reference. Object tracking approaches usually cannot cope with clutter and occlusion as moving objects or persons within the scene [11]. Other systems tend to jitter, because they apply fast 2D feature extraction methods to every single image [5,2], which can suffer from few features or poor feature localization and have to be regularized by temporal pose filtering. Our system overcomes these limitations. The key idea is that feature tracking is improved by compensating the features' appearances with respect to 3D viewpoint and lens effects, which can efficiently be done on graphics hardware with sub-pixel accuracy. The approach is separated into an offline and an online phase. During the offline phase, a very wide field of view camera (e.g. fish-eye with 180°) is moved within a scene and a structure-from-motion-approach [1] is applied to reconstruct the environment as a textured triangle mesh. Since no time-constraints are imposed during offline-processing, an optimal batch tracking with bundle adjustment and multi-camera depth estimation is possible, yielding high quality models. Next, robust 2D features (e.g. MSER [7]) are extracted from reference images, their 3D coordinates are computed and the features are stored in a database according to [4]. The textured triangle mesh and the robust features database serve as an offline reference model.

In the online phase, robust features are extracted from the first image and matched to the robust features of the offline database similar to [4]. Using these correspondences an approximate camera pose is estimated for initialization of the system. Our contribution here focuses on the subsequent tracking part of the online system: From the approximate pose, we synthesize a fish-eye image of our offline model using the same (intrinsic and extrinsic) camera parameters as the real fish-eye camera has (see section 2). We need to minimize the difference between rendered and real camera image to obtain the correct camera pose. However, since we want to cope with outliers and moving scene content, we do not use a direct gradient based approach to estimate the pose parameters as in [11] but search for local 2D offsets of individual free-form surfaces using the KLT [6]. From the exact locations of the surfaces in the camera image we can compute the final camera pose as described in section 3. Section 4 is dedicated to the evaluation of the system on real and synthetic data followed by a conclusion.

2 Spherical Camera

We propose to use a wide field-of-view camera, e.g. with a fish-eye lens, which has a nearly linear and isotropic relation between distance in pixels to the principal point and the angle between the ray and the optical axis[17]. Fleck [10] calls this the equidistant projection. A comparison between spherical and perspective cameras regarding tracking can be found in [9], who showed that pose estimation is more accurate with a wider field of view and that the lower angular resolution of the fish-eye lens is more than compensated by its wide field of view. Furthermore, such a camera covers a larger solid angle and therefore features can be seen for a longer period of time in image sequences. Let $P()$ be the function that computes a 2D image point x_i from a 3D scene point X_i, which takes care of all internal and external camera parameters of our real camera (CCD size, lens distortion, camera pose p, ...): $P(p, k, X_i) = x_i$

P is actually composed of extrinsic camera parameters, i.e. the pose p (position and orientation), and intrinsic camera parameters k, which describe the mapping of 3D points *in the camera coordinate system* to image coordinates. The internal parameters do not change, since they depend only on the lens and the hardware, we are going to estimate the pose. We describe the internal camera parameters with the function $K_k()$, where $K_k()$ maps projection rays in the camera coordinate system to 2D points in the image depending on a vector of internal parameters k. Therefore when we measure an image point x_i in any camera we can use K_k^{-1} to compute the ray that maps a 2D image point onto the unit sphere within the camera coordinate system. We define the mapping from world coordinates to normalized camera coordinates by $\hat{P}$: $\hat{P}(p, X_i) = K_k^{-1}(x_i) = \hat{x}_i$ where $\hat{P}$ is only a function of the pose and the 3D point. k can be determined by calibration [13]. If the effects of $K_k()$ are removed from the image measurement, we compute on rays in the camera coordinate system, which is easily applicable for all camera models with a single center of projection.

In a similar way we can synthesize fish-eye images using the graphics hardware: Given a camera position, we render 6 perspective views in all 6 directions (cube-mapping of environment). Afterwards we stitch these images together to form a fish-eye image (displacement mapping). This exploits again that for each pixel in the fish-eye image, we know the ray and therefore the coordinates where a perspective camera observes this ray. This can be efficiently implemented using OpenGL/CG and runs directly on the graphics hardware. Furthermore we combine the zBuffer values to produce a spherical depth map in a similar way.

3 Camera Tracking

We will first review the offline model generation process and the general system aspects, then we will study the online correspondence search and pose estimation.

3.1 Offline Model Generation

During the offline phase a video of the scene is captured systematically by scanning the possible range of viewpoints that will be used during online processing. In this way we *learn* a 3D reference model for later use. The intrinsics of the fish-eye camera are known [13], therefore coordinates in the image can be identified with rays in the camera coordinate system. First we perform a feature based reconstruction of the camera path similar to [1] using correspondences from the KLT tracker [6]. Next we generate depth maps by applying a cylindrical rectification method in 3D (again a way of abstracting from the underlying camera distortion) to the fish-eye images to use a standard stereo algorithm. The results are fused to robustify the depth maps (for example see figure 1). From these depth maps, free-form surface models can be built which are represented by textured triangle meshes. If a very fine resolution is chosen, the real surfaces

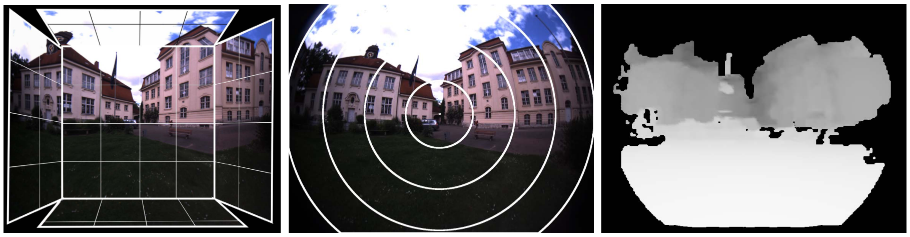

Fig. 1. Left: 3D view of cube-mapped environment (perspective views) Middle:Fish-eye image with center, 45°, 90°, 135° and 180° field of view circles, Right: Example of Reconstructed Fish-eye Depth Map

are approximated quite well at the cost of a huge number of triangles. If on the other hand a very coarse resolution is chosen, the free-form surfaces are actually approximated by only few planar patches (triangles), which can be rendered more efficiently. Those regions which have been tracked well over long time in the offline phase are obviously visible from several viewpoints and serve as a hint for the online phase where to register the triangle mesh with the camera image.

The textured triangle mesh is a reconstructive model of the scene: It represents the scene well in the sense that we can render a virtual view of the scene from any given viewpoint. However, when the camera is switched on during the online phase, no prior information is given about the viewpoint and we have to initially register the camera with respect to the scene, i.e. we also need a discriminative model. We solve this by creating a database of robust features (e.g. MSER[7]) as described in [4], which allows efficient recognition of scene parts.

3.2 Online Pose Estimation

Once we have set up the model we start the online phase, where the system registers against the database [4] and begins the tracking. The registration is robust and needs no approximate pose. However, the subsequent tracking approach is much faster and also more accurate, therefore the database is only used when the system is lost.

Correspondences between Image and Model. Given an approximate pose, the model is rendered and the true pose is computed from the displacement vectors between regions of the rendered image and the real camera image. By using a fish-eye lens we have all the advantages in visibility and geometrical stability, however the appearance of the model is quite different between distant camera poses. Therefore we need the rendering to undistort these effects by warping the model image into the new viewpoint and allowing to establish correspondences using standard techniques like KLT. After the free-form surfaces are rendered, we check geometrically which ones are projected into the virtual image and search for textured regions with a minimum size (e.g. 7x7 pixels) inside each free-form surface projection. We save its center point x_i and create its corresponding 3D

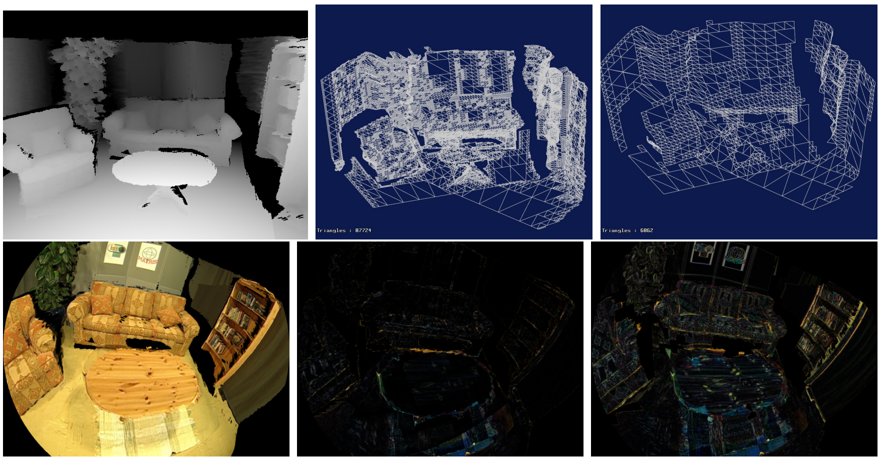

Fig. 2. From Left to Right, Top Row: Perspective Depth Map and Two Reduced Triangle Meshes (87724, 6882 triangles), Bottom Row: Ground Truth Fish-eye Image (left) and Photometric Difference for a sample view from meshes above

point X_i by back-projecting the viewing ray onto the model (using the depth from the renderer). This delivers a 3D model feature.

We start individual gradient-based minimizations [6] of the intensity differences at these locations x_i between the patches in synthesized and real image. This is more robust than a gradient-based global optimization of the pose across the whole image (as in [11]), since several scene parts may be occluded by persons or other unmodeled objects and it is hard to decide within one iteration step, which pixels should be used and which not. For a whole free-form surface we can test the projection error to see whether it is an outlier. Furthermore, the difference minimization is always carried out between a synthesized image and the actual camera image. This way, the offline model serves as a global reference and we will not accumulate drift as it would be the case when one tracks from camera image to camera image. The rendering can be seen as a fish-eye compensation of the patch for tracking. For simplicity, we use the standard KLT tracker, since our prediction (the rendered image) is usually very close. However, if illumination changes occur it is easy to use a more light insensitive version.

Robust Pose Estimation. The resulting 2D-3D correspondences are then processed in a robust non-linear pose estimator (M-Estimator with Huber error function $h()$ to limit the influence of mismatches), which starts at the predicted pose and minimizes the ray error for all 2D-3D correspondences. More precisely, the position x_i and the covariance matrix $C_{x_i x_i}$ from the KLT tracker in the original image are transformed to position $\hat{x}_i$ and a covariance $\hat{C}_i$ which is obtained through an unscented transform[15]. Now the Mahalanobis distance between $\hat{x}_i$

and the ray of the 3D point is minimized, where the transformed covariance $\hat{C}_i$ of the tracked point defines the Mahalanobis error metric.

$$\sum_i h\left((\hat{x}_i - \hat{P}(p, X_i))^T \hat{C}_i^{-1} (\hat{x}_i - \hat{P}(p, X_i))\right) \rightarrow min$$

We are looking for the pose p which minimizes the sum of these distances for all points. Once the pose is computed it is possible to render another fish-eye image from that pose and performing the KLT step again, this way iterating towards an even better pose. If and how many iterations are needed depends on the quality of the pose prediction and therefore mainly on the speed and smoothness of camera movement and the speed of computation. Within the camera movement, the rotation is the most critical part, because fast rotations change the fish-eye image more drastically than fast translations (when assuming a certain distance from the scene). In [14] it was found that the critical point is mainly the ability of the KLT tracker to establish the correspondence between rendered and real image at all and that no significant improvement could be observed when rendering more than two times.

4 Experiments

In this contribution the focus is on the analysis-by-synthesis part of the tracking, therefore we assume an approximate initialization in the following evaluation is given. We will compare the approach based on a synthetic scene for ground truth and a real outdoor scene.

4.1 Ground Truth Experiments

As synthetic ground truth we used a model of a real living room scene with real textures as reconstructed by the offline modeling part (see figure 2). The model is fused from four perspective depth maps of the scene, consists of 1.2 million triangles (the bounding box is about 5m x 4m x 2m) and we generated a sequence of 350 fish-eye images (140° field of view, camera translation about 1.5m, rotation in all directions, where the vertical axis rotations dominate by up to 80°) with ground truth pose information.

Sensitivity to Model Accuracy. The accuracy of the pose estimation depends on the goodness of the model used for tracking. Therefore, we compared rendering speed and average pose estimation accuracy (figure 3) at varying resolutions of the triangle mesh for tracking (figure 2). The number of triangles is reduced from 1.200.000 down to 1.600 by a combination of depth map resolution reduction and quad tessellation similar to what has been proposed in [12].

The main result of this evaluation is not surprising: With increasing number of triangles rendering performance goes down; if the GPU resource limit is reached, real-time tracking becomes infeasible. The pose estimation error decreases about logarithmically with increasing number of triangles. In the extreme case of only a

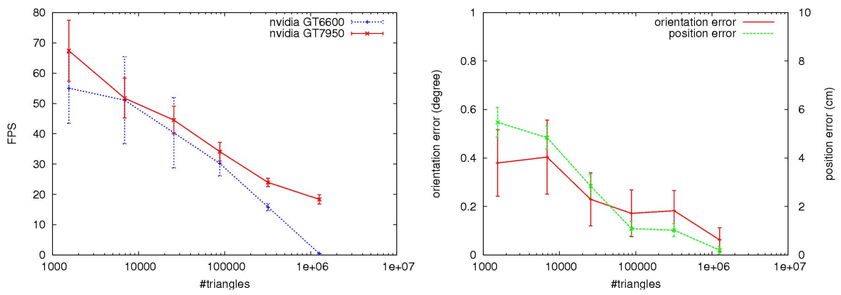

Fig. 3. Model detail in number of triangles. Left: Pure Rendering Frame-rate, Right: Average Orientation and Position Error on Living Room Sequence.

few triangles the scene is actually represented by planar patches, which showed to be only usable as long as the underlying scene is planar. Otherwise the rendering does not fulfill the undistortion goal: the rendered and the camera image look significantly different and cannot be matched by the KLT tracker. Only those points are found which actually do lie on planes and are approximated well.

Comparison of different algorithms. The system has been compared against a) incremental structure-from-motion using the same camera (140° FOV) but no model and b) model-based tracking with a 40° FOV perspective camera with the same number of pixels and same number of surfaces (of which the perspective camera sees not all at once). Pose error is given as position (translational) error in cm and orientation error in $degree$ (axis-angle representation of the rotation between the ground truth camera and the estimated camera). The sequence is run forward and backward, generating a total of 700 frames with image 1 and 700 at identical pose. The results in figure 4 show that the model-based fish-eye tracking outperforms both other approaches. The error is constantly low over the complete sequence (average errors: position: 0.3 cm, orientation: 0.1°).

The structure-from-motion algorithm a) has no prior model and generates the model *on the fly*. Therefore, the average pose error is higher than with the model. Scale was fixed such that the tracking can be compared with the model-based approaches. Drift does not accumulate very much since all features are visible in most images, however we see an error increase as the camera moves away from the initial position. We deliberately left out the bundle adjustment, which would clearly help, but which is not feasible in real-time applications. Average position error is about 2 cm, average orientation error 0.3°.

The perspective model-based tracking b) on the other hand has difficulties in distinguishing between camera rotation and camera translation, which can be seen from the high correlation between orientation and translation error in figure 4. Furthermore it does only see about 100 of the about 500 free-form surfaces in the model at a given time because of its limited field of view. The errors are much higher with average position error 4 cm and orientation error 0.8°.

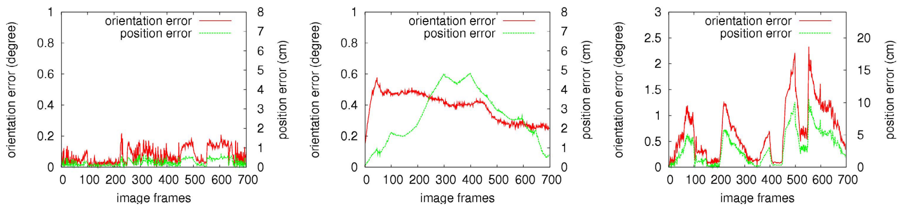

Fig. 4. Algorithm comparison on ground truth living room sequence (350 images forward+backward). Left: Fish-eye Model Tracking, Center: Fish-eye Structure-From-Motion, Right: Perspective Model Tracking (40° FOV) with triple error range.

Real outdoor fish-eye sequence. To prove the applicability of our method we evaluate a real sequence (see also figure 1), which consists of 1400 images of 1200×1600pixels, that were taken with a fish-eye lens covering a viewing angle of 185°. The camera was moved handheld and translated approximately 6m sidewards while panning up to 90°. The filmed buildings were up to 20m away and 12m in height. The camera path was reconstructed with the structure from motion approach using the full fish-eye images as explained earlier. The resulting depth map was used to create a mesh yielding a 3D model of the scene which consists of 90303 triangles (compare figure 5, left).

Without ground-truth data, the verification of the estimated camera path is difficult. One way to check for consistent model and camera path reconstruction is to augment the model into a sequence. The right image of figure 5 shows an augmentation of the model rendered with the estimated camera parameters. In order to provide an augmentation which is distinguishable from the background image, the texture of the model was replaced by its gradient magnitude. While evaluating the model tracking, the difference images between the original image and the rendered model view were monitored. This qualitative evaluation showed that the observable tracking error was in the range of one pixel.

In order to analyze potential accumulation of errors in pose estimation for long sequences, 360 consecutive images of the real sequence were processed forward and backwards several times, starting at the middle of the sequence. The central image position is reached eight times and compared to the first pose, which should always be the same. Figure 5 shows the extent of this path which is approximately 2 meters to the left and to the right of the middle camera (green). Looping through this sequence resulted in 2160 images for tracking. Given the pose for the first (central) image, the camera poses for this "oscillating" sequence are estimated using SfM tracking and model based tracking with 400 features for both. Model based tracking uses only one rendering iteration.

Table 1 (top two rows) compares the error development at the middle image over consecutive passes of a looped sequence using tracking on fish-eye images, but without a model. Although the error is not constantly growing with each pass an error increase is visible. On the other hand the tracking error observed

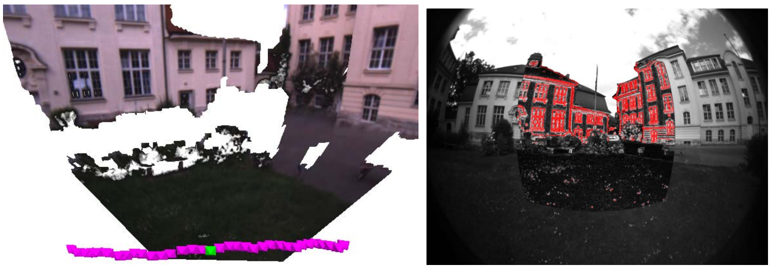

Fig. 5. Left: Perspective view on reference model and extent of camera movement used for drift measurement. The camera is going forward/backward from one end of this path to the other. It passes the green middle camera, where the pose estimation is compared to previous and following passes (see table 1). Right: augmentation of model view and real image using estimated parameters. The texture used for the augmentation is the gradient magnitude of original texture, strong gradient edges colored in red.

Table 1. Pose error evaluation for a looped image sequence, which passed the image under inspection eight times. The SfM rows show the position and orientation error using a structure from motion based tracking and how pose estimation has drifted when passing this image. The model rows prove the avoidance of error accumulation when tracking is supported by a model.

	pass 1	pass 2	pass 3	pass 4	pass 5	pass 6	pass 7	pass 8
ΔT SfM	2.57 cm	1.92 cm	1.92 cm	2.79 cm	2.41 cm	1.06 cm	3.53 cm	3.22 cm
$\Delta\phi$ SfM	0.098°	0.085°	0.085°	0.11°	0.11°	0.02°	0.13°	0.14°
ΔT Model	0.73 cm	0.82 cm	0.69 cm	0.73 cm	0.84 cm	0.68 cm	0.73 cm	0.81 cm
$\Delta\phi$ Model	0.047°	0.047°	0.046°	0.047°	0.048°	0.046°	0.047°	0.047°

using the model (last two rows) is confined and does not increase over consecutive passes. This confirms that the system does not drift. Furthermore the pose error is smaller at all times when compared with the SfM tracking.

5 Conclusion

We have discussed a camera tracking system, which first builds a textured model from the environment and afterwards uses the model in an analysis-by-synthesis approach for tracking. The graphics hardware is exploited to render a distortion-compensated and perspectively warped model image with an approximate pose. Since this compensates the effects of the wide-angle lens, now full advantage can be taken of the fish-eye properties, which proved to be superior to perspective cameras in tracking. It was shown that there is no drift accumulation over time and therefore the system is well-suited to work on infinitely long image sequences. The accuracy of the model approximation should fit well the free-form surfaces, since planar approximations of curved surfaces degrade the accuracy. On current

GPUs a model complexity of about 100.000 triangles is feasible. We showed the applicability of the approach by using a model of a real outdoor building and a semi-artificial living room sequence.

References

1. Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. IJCV 59(3), 207–232 (2004)
2. Lepetit, V., Lagger, P., Fua, P.: Randomized Trees for Real-Time Keypoint Recognition. In: CVPR 2005, San Diego, CA (June 2005)
3. Thomas, G.A., Jin, J., Niblett, T., Urquhart, C.: A Versatile Camera Position Measurement System for Virtual Reality TV Production. In: Proceedings of International Broadcasting Convention, Amsterdam, pp. 284–289 (1997)
4. Koeser, K., Haertel, V., Koch, R.: Robust Feature Representation for Efficient Camera Registration. In: Franke, K., Müller, K.-R., Nickolay, B., Schäfer, R. (eds.) Pattern Recognition. LNCS, vol. 4174, pp. 739–749. Springer, Heidelberg (2006)
5. Skrypnyk, I., Lowe, D.G.: Scene Modelling, Recognition and Tracking with Invariant Image Features. In: ISMAR 2004, Arlington, pp. 110–119 (2004)
6. Tomasi, C., Kanade, T.: Detection and Tracking of Point Features, Carnegie Mellon University, Technical Report, CMU-CS-91-132 (April 1991)
7. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust Wide baseline Stereo from Maximally Stable Extremal Regions. In: Proceedings of BMVC02 (2002)
8. Davison, A.J.: Real-Time Simultaneous Localisation and Mapping with a Single Camera. In: Proceedings International Conference Computer Vision, Nice (2003)
9. Streckel, B., Koch, R.: Lens Model Selection for Visual Tracking. In: Kropatsch, W.G., Sablatnig, R., Hanbury, A. (eds.) Pattern Recognition. LNCS, vol. 3663, pp. 41–48. Springer, Heidelberg (2005)
10. Fleck, M.: Perspective Projection: The Wrong Imaging Model. TR 95-01, Computer Science, University of Iowa (1995)
11. Koch, R.: Dynamic 3D Scene Analysis through Synthesis Feedback Control. IEEE Transactions PAMI 15(6), 556–568 (1993)
12. Evers-Senne, J., Koch, R.: Image Based Rendering from Handheld Cameras using Quad Primitives. In: Evers-Senne, J., Koch, R. (eds.) VMV 2003, November 2003, Munich, Germany (2003)
13. Scaramuzza, D., Martinelli, A., Siegwart, R.: A Toolbox for Easy Calibrating Omnidirectional Cameras. In: Proceedings of IROS 2006, October 2006, Beijing, China (2006)
14. Koeser, K., Bartczak, B., Koch, R.: Drift-free Pose Estimation with Hemispherical Cameras. In: Proceedings of CVMP 2006, November 2006, London (2006)
15. Julier, S., Uhlmann, J.: A new extension of the Kalman filter to nonlinear systems. In: Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando (1997)
16. Bleser, G., Wuest, H., Stricker, D.: Online Camera Pose Estimation in Partially Known and Dynamic Scenes. In: ISMAR 2006, Los Alamitos, California, pp. 56–65 (2006)
17. Neumann, J., Fermuller, C., Aloimonos, Y.: Eyes from eyes: New cameras for structure from motion. In: IEEE Workshop on Omnidirectional Vision, pp. 19–26 (2002)
18. Micusik, B., Pajdla, T.: Structure From Motion with Wide Circular Field of View Cameras. IEEE Transactions on PAMI 28(7), 1135–1149 (2006)