View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Crossref

ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-3/W2, 2013
ISA13 - The ISPRS Workshop on Image Sequence Analysis 2013, 11 November 2013, Antalya, Turkey

SELF-LOCALIZATION OF A MULTI-FISHEYE CAMERA BASED AUGMENTED REALITY SYSTEM IN TEXTURELESS 3D BUILDING MODELS

S. Urban, J. Leitloff, S. Wursthorn and S. Hinz

Institute of Photogrammetry and Remote Sensing (IPF),
Karlsruhe Institute of Technology (KIT)
Englerstr. 7, D-76137 Karlsruhe, Germany
(steffen.urban, jens.leitloff, sven.wursthorn, stefan.hinz)@kit.edu
http://www.ipf.kit.edu

Commission III WG III/3

KEY WORDS: computer vision, fisheye camera, real-time object detection, model-based tracking, augmented reality

ABSTRACT:

Georeferenced images help planners to compare and document the progress of underground construction sites. As underground positioning can not rely on GPS/GNSS, we introduce a solely vision based localization method, that makes use of a textureless 3D CAD model of the construction site. In our analysis-by-synthesis approach, depth and normal fisheye images are rendered from presampled positions and gradient orientations are extracted to build a high dimensional synthetic feature space. Acquired camera images are then matched to those features by using a robust distance metric and fast nearest neighbor search. In this manner, initial poses can be obtained on a laptop in real-time using concurrent processing and the graphics processing unit.

1 INTRODUCTION

This paper describes a multi fisheye camera system to support localization of a mobile augmented reality (AR) client on subway construction sites. It is part of the work of a joint research group on computer-aided collaborative subway track planning in multi-scale 3D city and building models. It addresses the challenges of integrating the collaborative planning effort within a synchronous software workflow for seamlessly modeling, managing and analyzing both CAD and GIS based 3D models (Breunig et al., 2011). A mobile client is supposed to support the planning process by augmenting camera images with 3D models of distinct construction stages on-site. Consequently, the camera snapshots need to be localized and orientated within the spatial context of the 3D model. The overlay of projected 3D data on these camera snapshots is supposed to help planers to compare planning stages with the actual status (Breunig et al., 2012).

The interior orientation of three cameras with fixed focus fisheye lenses is determined in a calibration step and is considered constant for all images grabbed with a particular camera. The six degrees of freedom of the exterior camera orientation need to be determined with every new image. A major challenge is the lack of GNSS signals at subway construction sites, which would ease getting at least an initial position and some stabilization points that are needed for a continuous tracking. The process of determining the six degrees of freedom of the camera's exterior orientation within the given 3D model with feature correspondences extracted both from the images and the untextured 3D models alone is the purpose of our sub-project within the research group.

We do not use depth images from stereo or range imaging devices due to their limited range and field of view (Weinmann et al., 2011). Cameras do easily fit on a building-site helmet in contrast to e.q. laser scanning devices. In this paper we will concentrate on the initialization step of our designated tracking process. We use gradient orientations, extracted from rendered images of the 3D model and compare these to gradient orientations in the real camera images by using a robust distance metric, to find the camera pose in the model.

1.1 Previous Research

The fact, that only an unknown fraction of the building model will be visible on the camera images is challenging and has rarely been addressed in other publications on model based tracking. In (David et al., 2003, David et al., 2004, Wuest et al., 2005) edge based methods compare 3D model edges projected into the image with regions that have high image gradients. (Drummond and Cipolla, 2002) project 3D points that lie on model edges onto the image plane and minimize distances to the nearest edges in the image. The matched 3D-2D point correspondences are used to solve the registration of the image within the 3D model. The system is supposed to operate in real-time, but a good initial registration is needed in order to minimize the distances. (Behringer et al., 2002), (Reitmayr and Drummond, 2006) and (Simon, 2011) need additional sensors for position (GPS) and orientation to initialize their tracking systems in large outdoor environments.
(Wuest et al., 2007) propose the use of an inertial measurement unit in the future for initialization.

(Ulrich et al., 2009) use offline training of various possible object poses to achieve an image based initialization. (Ottlik and Nagel, 2007) show automatic initialization in the domain of vehicle tracking in traffic videos. Both implementations are robust against occlusions but the models are supposed to completely fit into the images.

In the remainder of this paper the multi fisheye camera system is introduced first as well as the calibration of the camera and the system. Afterwards the self-localization method is shown and each step in the process is detailed. Subsequently a test on simulated data is carried out, to adjust parameters that are crucial to the initialization and to evaluate pose estimates. Finally we give a conclusion and describe our future work.

2 CAMERA SYSTEM

The camera system consists of three synchronously triggered cameras, equipped with equal fisheye lenses with a FOV of $\sim 190^\circ$.

Fisheye cameras are often superior to narrow angle perspective cameras in tracking tasks (Svoboda, 1998, Micušık and Pajdla, 2006). Features stay longer in the field-of-view during camera movement, partially occluded model parts occur only in small image regions and pose estimation is often better conditioned. To further improve the quality, stabilize the pose estimation and provide a maximum number of visible features, three fisheye cameras are used. The horizontal angle between each principal axis averages 120°, creating a 360° panoramic view of the environment. This particular camera arrangement was chosen to allow a robust localization even if one or two cameras fail to maintain their pose estimate, which could happen on construction sites. (Fig. 1).

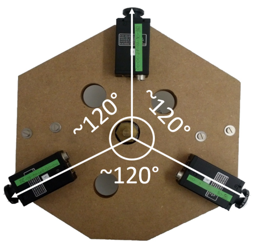


Figure 1: System configuration of the three fisheye cameras.

2.1 Fisheye Camera Model and Calibration

Camera calibration is essential to many vision tasks. The result is a mapping function that describes the relation between a pixel on the image plane to its corresponding 3D ray. In an analysis-by-synthesis tracking approach, the calibration is crucial to the rendering step, as the virtual camera needs to fulfill the same mapping functions as the real camera. The calibration of a fisheye camera differs from perspective camera calibration, because camera models need to allow for incident angles greater 90°. In recent years, different models and calibration methods for fisheye cameras have been published (Scaramuzza et al., 2006, Mei and Rives, 2007, Micušık, 2004). A comprehensive overview is given in (Puig et al., 2012). We chose the unified taylor model introduced by (Scaramuzza et al., 2006). The following section details parts of the model that are important for the rendering (Sec. 3.2). Let $P = (X, Y, Z)^T$ be a point in world coordinates. Then the function $f(u, v)$, that maps the 2D image point $p = (u, v)^T$ to P, i.e. the forward-projection, is calibrated:

$$\boldsymbol{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} u \\ v \\ f(u,v) \end{bmatrix} \tag{1}$$

The fisheye lens is assumed to be rotational symmetric thus the polynomial can be written as:

$$f(\rho) = a_0 + a_2\rho^2 + ... + a_n\rho^n \tag{2}$$

with

$$\rho = \sqrt{u^2 + v^2} \tag{3}$$

We chose a polynomial degree of $n = 4$ in this paper, which gave the best calibration results. During rendering, the mapping function of a 3D object point onto the image plane is needed. This back-projection is modeled by the inverse of $f(\rho)$:

$$d = f^{-1}(\Theta) \tag{4}$$

where d is the distance of a back projected point from the image center and

$$\Theta = \arctan\left(\frac{Z^2}{X^2+Y^2}\right) \tag{5}$$

is the incident angle to the principal axis. Thus temporary image coordinates can be calculated:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} X\frac{f^{-1}(\Theta)}{\sqrt{(X^2+Y^2)}} \\ Y\frac{f^{-1}(\Theta)}{\sqrt{(X^2+Y^2)}} \end{bmatrix} \tag{6}$$

Then a affine mapping is applied to compensate for small distortions, non-square pixels and misalignments between lens and image plane axis. Finally the image coordinates are related to the upper left image corner by adding the coordinates of the principal point:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{7}$$

with principal point x_c, y_c and affine parameters c, d, e.

2.2 System Calibration

After determination of each cameras intrinsic parameters, the multiple fisheye camera configuration can be examined. In order to fully exploit the configuration, relative poses between each camera are necessary. In this way, the systems exterior orientation can be maintained, even if two cameras fail to estimate their pose. As each camera points in a different direction, a calibration object that is only visible in one camera at a time, like a calibration pattern, is unusable. Hence we directly made use of the 3D building model in which the system is moving (for a description of the model see section 4). A trajectory was recorded and synchronous keyframes from this trajectory were selected. Then 2D image features and their corresponding 3D coordinates in the model were manually extracted. Next, the exterior orientation for each keyframe and each camera was estimated from these 2D-3D correspondences. The relative poses were then extracted from a series of exterior orientations for each camera.

3 SELF-LOCALIZATION

The system is supposed to estimate its initial position without human intervention. This process is detailed in Figure 2. It can be divided into two parts, i.e. online and offline phase. During the *offline* phase, the system creates virtual camera poses, renders synthetic views of the model and extracts features from them. These features are then clustered and saved. In the *online* phase the cameras acquire images and extract feature vectors from them likewise. These features are then matched to the synthetic features for each frame and each camera with a particular metric. If the metric exceeds a certain threshold the virtual camera pose is assigned to the real camera and the system has localized itself in the model. The following sections detail each consecutive step shown in Figure 2.

3.1 Position Sampling

At first glance the camera pose in the 3D model seems to be entirely random. This would require a huge amount of pre-sampled positions and hence to many feature vectors to process. But since the system is supposed to be head mounted, some constraints can be applied.

A sampled position is expressed in spherical coordinates. The origin of the sphere is at position X, Y, Z in model coordinates

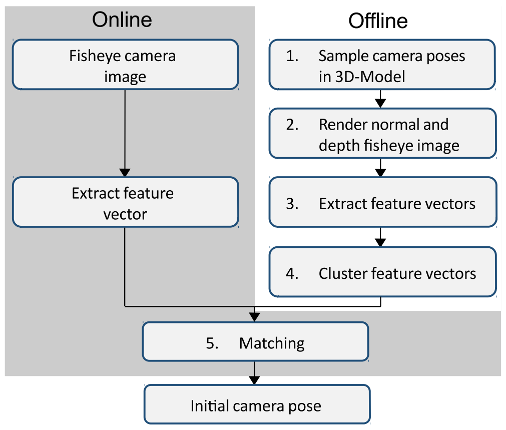
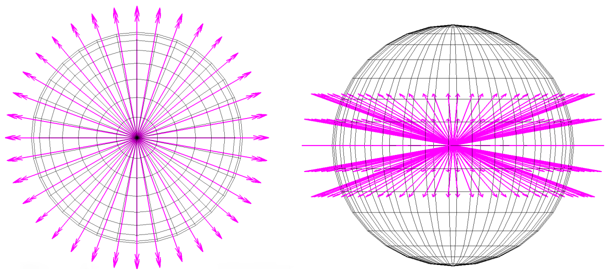

Online
Fisheye camera image
Extract feature vector
Offline
1. Sample camera poses in 3D-Model
2. Render normal and depth fisheye image
3. Extract feature vectors
4. Cluster feature vectors
5. Matching
Initial camera pose

Figure 2: Flow-chart of the self-localization process.

and the view direction is expressed by a polar (or yaw) angle ϕ and a azimuth (or pitch) angle θ. Now assumptions about the range of values can be made. For instance, the operator tilts his head in a narrow range which constrains the pitch angle θ. In this work θ is set to $[-20^\circ, 20^\circ]$ from the horizon and is sampled in 10° steps. The yaw angle is free from $[0, 360^\circ]$ and is sampled in 10° steps in this example. The horizontal sampling is a parameter that can be optimized, though, because of the similarity of adjacent views (Fig. 3). Further the operators body height is known



(a) view onto x-y plane (b) view onto y-z plane

Figure 3: (a) Camera view directions in the x-y plane. Depicted is the sampling of the polar angle. (b) Camera view directions in the y-z plane. Depicted is the sampling of the azimuth angle. This covers approx. the range an operator would tilt his head.

and thus the Z component can be set accordingly. This still leaves the X and Y component free. To further reduce the number of presampled positions, X and Y are sampled in a rectangular grid at level Z (Fig. 4). The spacing between each position determines the density of the position sampling. We chose a grid density of 0.5 m for this work. Sampling the view sphere also prevents from presampling positions for each camera separately since the distance from each camera to the origin is small w.r.t the density of the grid.

3.2 Rendering

After sampling virtual camera poses in the 3D model, synthetic views of the 3D model are generated (Fig. 5). This approach is called analysis-by-synthesis (Wuest et al., 2007). The idea is, to render a view of the 3D model, extract all visible edges from it and match them to the real camera image of the model. OpenGL (Khronos Group, 2013) is used to perform hidden line removal and to render the image in a normal and a depth image. This two-time rendering is necessary, as the normal image contains all crease edges and the depth image all step edges. A conventional rendering pipeline only supports two kinds of projection matrices, either orthographic or perspective. Neither applies to fisheye cameras.

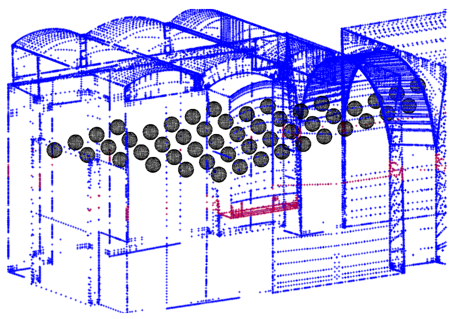
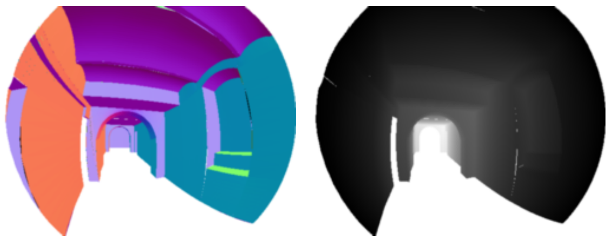


Figure 4: Distribution of view speres in the horizontal grid. The z-coordinate of the grid resamples approx. the height of the operators head.

Thus we elude the fixed vertex processing by using a custom vertex shader. Latter allows us to manipulate the geometry of the scene. First, the model is transformed to the camera reference frame using the model-view matrix, which contains the camera pose in the 3D model. Now the incident angle Θ for each vertex can be calculated Eq. (5) and the inverse mapping Eq. (4) for each viewing ray is determined. Then the corresponding pixel is calculated according to Eq. (6), Eq. (7). Hereby the calibrated mapping function of each real camera is exploited in a vertex shader, to extend the simple perspective camera view to a geometrically correct fisheye view.

After vertex processing, a fragment shader is used to color the pixels according to their normal (Fig. 5(a)) and depth (Fig. 5(b)) information respectively. At this point it is important to note, that the normal and depth values are independent of the vertex transformation. The z-value of each vertex is untouched during the mapping as well as the normal vectors. Latter are assigned to each face and vertex respectively and are calculated prior to the mapping. Otherwise the fisheye mapping would change these information and the resulting image would contain wrong normal and depth values.



(a) normal image (b) depth image

Figure 5: Rendered fisheye images for combined crease edge (a) and step edge (b) extraction.

Image pyramids help to speed up feature extraction, but the process of generating these pyramids for both images, i.e. normal and depth images, slows down the overall process. Thus, instead of rendering a full resolution image followed by subsequent downsampling, we chose to render different image octaves directly by applying different viewport sizes, followed by concurrent edge extraction.

3.3 Feature Extraction

Until now, each virtual camera position consists of two images. In this step the information of both images is combined and 1D feature vectors are created. The extraction of gradient orientations is similar to (Hinterstoisser et al., 2012). First the x- and y-derivatives of both images are extracted for each channel using a 3×3 sobel kernel without gaussian smoothing. The smoothing step can be omitted since rendered images do usually not contain noise. Then a gradient magnitude image $I_C = \sqrt{G_{C,y}^2 + G_{C,x}^2}$ is calculated for each channel, where $G_{C,x}$ and $G_{C,y}$ are the sobel derivatives in x and y direction for a channel $C \in \{R, G, B, D\}$, where the first three belong to the normal and the last to the depth image.

All magnitude arising from the normal image are tested against a threshold. Latter corresponds to a orientation change. Round structures and surfaces in 3D models are appoximated by many faces. The edge extraction would yield all edges, even if they are not present in the real image. To avoid this, the threshold has to be set accordingly. If no edge has been detected for a particular pixel in the normal image, the depth image is tested. In this case another threshold needs to be set, which corresponds to the z-buffer range and precision. We then choose the highest magnitude $\hat{G}$ of all magnitude images for a particular pixel u, v and extract its orientation

$$\Phi(u,v) = \arctan\left(\frac{\hat{G}_{C,y}(u,v)}{\hat{G}_{C,x}(u,v)}\right) \tag{8}$$

The combined gradient image is depicted in Figure 6(a). Edges are not thinned using non maximum suppression. However it showed, that broader edges lead to better matching results, as there will never be a perfect match and thus broader edges serve indirectly as a pull-in range for the nearest neighbor matching.

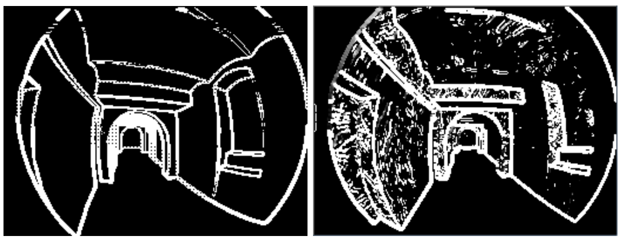

(a) combined edge image (b) synthetic image edges

Figure 6: (a) Combined edge image. After extracting edges from normal and a depth image seperately, the maximum gradient is chosen to extract edge orientations. (b) Extracted edges from rendered model. The added noise to pose and image, respectively is visible.

Each combined edge image is then reshaped to a 1D holistic feature vector that contains the orientations of all visible edges. In our implementation the feature extraction is carried out on the 4th pyramid level. Here the images size is 49×65 pixels leading to a feature dimension of 3185.

3.4 Feature Matching

The last step to self-localization is crucial. The offline created, synthetic feature vectors have to be matched against the incoming camera images, using a robust distance metric. Since a linear search trough all features for an exact match would not be real-time capable, an efficient way of clustering and searching a nearest neighbor to a given feature vector is used.

3.4.1 Distance Metric The matching is solely performed on edges. Hence the distance metric has to be robust to occlusion, illumination change and clutter. Geometric deformations can be neglected since we use a rigid 3D model. We use the metric defined in (Hinterstoisser et al., 2012) which is adapted from (Steger, 2002). Latter shows its robustness to occlusions and illumination changes by taking the absolute value of the cosine: Unsimilar egdes contribute little to the sum and negative values, caused by different illuminations, are mapped to the positive range. The metric is based on gradient orientations to determine a measure of similarity. In contrast to (Steger, 2002), Eq. (9) minimizes the number of features and calculations during tree construction and matching:

$$d = \sum_{i=1}^{w \cdot h} |\cos(\Phi_{s_i} - \Phi_{t_i})| \tag{9}$$

where Φ_{s_i} and Φ_{r_i} are gradient orientations of search and template image and w, h are width and height of the search image.

3.4.2 Clustering and Nearest Neighbor Search As presented in Section 3.1 a lot of virtual cameras are sampled and high dimensional feature vectors are extracted. For instance, 50 presampled positions with an angular increment of 10° lead to 9000 feature vectors. Taking the feature dimension of 3185 from the last section, this would require $\sim$ 30 million calculations in a linear search, as Eq. (9) had to be computed for each template image against the search image.

Hence, we experimented with multiple randomized kd-trees and hierarchical trees with k-means clustering and use the implementation of flann (Muja and Lowe, 2009) a library for fast approximate nearest neighbor search to high-dimensional feature vectors. Usually the performance of nearest neighbor search decreases with the dimensionality of the feature vector. But (Muja and Lowe, 2009) show that for real world data sets, where strong correlation between dimensions exist, the precision remains almost constant. Both trees have different parameters which are crucial relating to speed and precision. The next section details the algorithm choice.

4 RESULTS AND DISCUSSION

For all our experiments, a L-shaped part of the buildings basement is used, which is characterized by bad light conditions, partially narrow walls and objects occluding edges. The 3D model is the result of a laser scan point cloud. The point cloud was simplified, cleaned and meshed. The final model has ~ 550.000 vertices and ~ 190.000 faces and consists of flat and round surfaces.

4.1 Simulated Data

To test our method and adjust parameters, i.e. the number of presampled positions, sampling density, edge thresholds, speed and search parameters the 3D model was rendered from 100 perturbed poses. These poses are randomly drawn from the set of all 9000 presampled poses. To simulate false and missing edges and to add image noise that occurs in a real world scenario, we attach a noisy, coarse resolution wall texture to the model and render it. The edges of the resulting image are depicted in Figure 6(b).

First, the linear search is applied to find the corresponding feature vector from all prerendered poses, as it is the most accurate method. We then add an increasing gaussian noise to position and angle values respectively. The noise is added separately to position and angle to test the robustness of the method against both parameters. The result is depicted in Figure 7. One can see that, for increasing angle noise, the accuracy reaches 50% when the noise reaches 5°. This corresponds exactly to half of the horizontal sampling distance we applied in Section 3.1. The position accuracy remains good up to a position noise of 0.15m. This is one third of the sampling density from Section 3.1. As the method is supposed to initialize the position in real-time, the

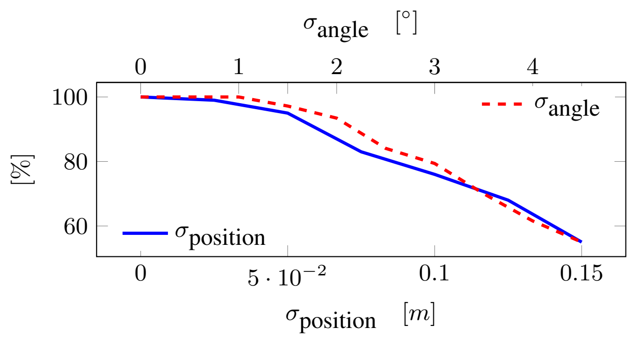


Figure 7: Precision of matching with increasing noise.

search speed is crucial. The linear search for 9000 features takes about 370ms on a Intel i7-3630QM CPU, which is not sufficient for real-time applications. Hence different trees and parameter combinations for flann (Muja and Lowe, 2009) were tested, to evaluate the accuracy against the linear search.

Again 100 textured images are rendered from perturbed positions. Then the 9 different trees from Figure 8(a) are searched. This time however, the number of recursively searched leaves is increased. This leads to a better search precision on the one hand, but reduces the search speed on the other. Thus, a trade-off between search speed, precision and tree build time has to be found. Figure 8 shows search speed and corresponding precision for an increasing search depth. For all six hierarchical k-means trees the search time increases faster than the precision. Considering that for real-time AR-applications all calculations for each camera have to be finished within 45ms $\approx$ 22fps (fisheye camera fps), the k-means tree is only feasible for lower search depths. However at a search depth of 300, the precision hardly exceeds 50%. A higher number of iterations during build time resulted in a longer build time but did not significantly improve the precision.

Hence we chose the randomized kd-tree. Even for 500 trees the build time is feasible and the precision for the simulated scenario keeps up with the hierachical k-means trees. The huge advantage is the search speed which is feasible for real-time AR.

4.2 Real Data

After adjusting parameters during the simulation, we tested our system under real conditions. To exploit the full spherical view of the scene, we create panoramic feature vectors, i.e. we stack the feature vectors of all three cameras together. We then match those to prerendered panoramic features from the feature creation step. The position sampling scheme is basically the same, but instead of rendering one image per sample position, we render three images, using the known relative poses of the cameras.

Since we did not yet acquire ground truth for each frame of an entire full frame rate trajectory, we conducted a qualitative visual verification of the initialization results for the 14 keyframes from Section 2.2 where the exterior orientation of the system was determined with a total station. We found that the initialization was correct in 11 out of the 14 keyframes. Figure 9 depicts such a case (and comprises a link to the sequence). The correctly matched model edges are overlaid in green. Note that the model edges do not fit perfectly. On the one hand we initialized the system from discrete poses, but in reality the trajectory is continuous, i.e. we initialized the nearest neighbor of the true pose. On the other hand the initialization is carried out on the lowest scale level. From here on, the pose has to be refined by, e.g. minimizing the squared distances to the nearest edge on each scale level.

5 CONCLUSIONS AND FUTURE WORK

In this paper, a novel approach to image based self-localization of a multi fisheye camera system in texture-less 3D models was presented. To approach the problem of a comparatively big model and the fact, that the camera pose is inside the model and only parts of it are visible at a current pose, virtual camera poses are presampled and fisheye camera views are rendered using OpenGL and a custom vertex shader. Then feature vectors based on gradient orientations are extracted on low pyramid levels and clustered for fast nearest neighbor search. A robust distance metric was used to match features during nearest neighbor search, which handles occlusions, clutter, illumination and contrast changes.

To further increase search precision and search speed for future work, redundancies between feature vectors have to be eliminated, i.e. two adjacent camera poses that contain approximately the same information can be omitted. In addition, all feature vectors live in memory during runtime and a decrease to the essential number could increase the scalability, however the presented configuration allocates 110MB ($9000 \times 3185 \times 4$ byte) memory which is viable even for larger feature sets.

One drawback of the method presented is, that it depends on a threshold to declare an initialization correct. The nearest neighbor search gives the best (highest distance metric sum) result for a given frame, which is not instantaneously correct and there might be other neighbors that are almost equally good and thus could be probably correct. Thus, in future work, we will base our method on a particle filter. Each presampled positions can be seen of as a particle and the distance metric can be used as a likelihood measure for each consecutive frame. This means, that all particles will have a high likelihood in the neighborhood of the correct pose, which can be described by a probability density function over the state space (presampled poses).

REFERENCES

Behringer, R., Park, J. and Sundareswaran, V., 2002. Model-based visual tracking for outdoor augmented reality applications. In: International Symposium on Mixed and Augmented Reality (ISMAR), Darmstadt, pp. 277–278.

Breunig, M., Borrmann, A., Rank, E., Schilcher, M. and Hinz, S., 2012. Towards 3d geoinformatics and computational civil engineering support for cooperative tracks planning. In: Proceedings of FIG Working Week, Rome.

Breunig, M., Rank, E., Schilcher, M., Borrmann, A., Hinz, S., Mundani, R., Ji, Y., Menninghaus, M., Donaubauer, A., Steuer, H. et al., 2011. Towards computer-aided collaborative subway track planning in multi-scale 3d city and building models. In: Proceedings of the 6th 3D geoinfo conference, Vol. 17.

David, P., Dementhon, D., Duraiswami, R. and Samet, H., 2003. Simultaneous pose and correspondence determination using line features. In: IEEE Conference on Computer Vision and Pattern Recognition, Madison, USA, pp. 424–431.

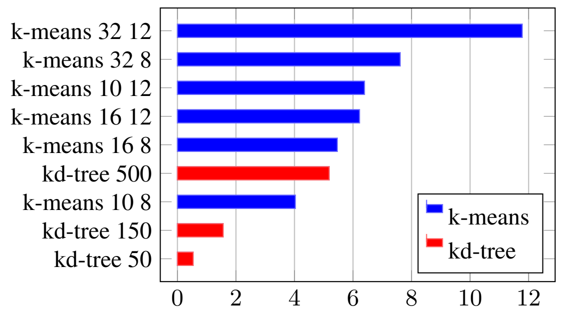
David, P., Dementhon, D., Duraiswami, R. and Samet, H., 2004. Softposit: Simultaneous pose and correspondence determination. International Journal of Computer Vision 59(3), pp. 259–284.

Drummond, T. and Cipolla, R., 2002. Real-time visual tracking of complex structures. IEEE Transaction on Pattern Recognition and Machine Intelligence 24(7), pp. 932–946.

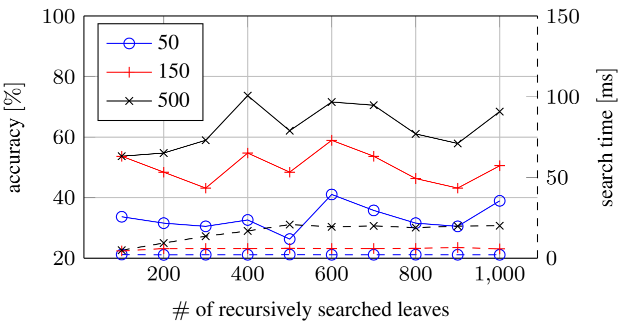
Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P. and Lepetit, V., 2012. Gradient response maps for real-time detection of textureless objects. Pattern Analysis and Machine Intelligence, IEEE Transactions on 34(5), pp. 876–888.

Khronos Group, 2013. Opengl - the industry standard for high performance graphics. `http://www.opengl.org`, visited in june 2013.

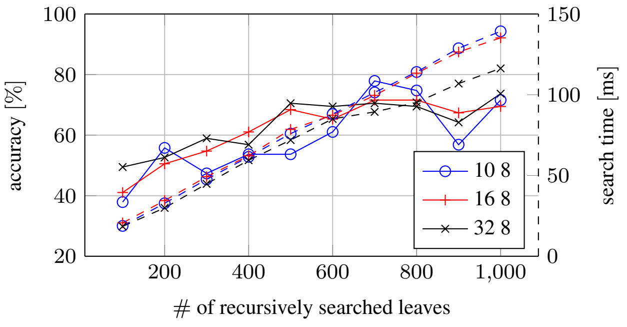
Mei, C. and Rives, P., 2007. Single view point omnidirectional camera calibration from planar grids. In: IEEE International Conference on Robotics and Automation.



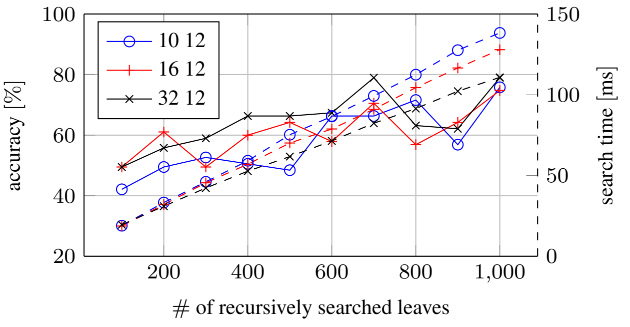
(a) comparison of build times for 9000 features in minutes



(b) kd-tree with randomized trees



(c) hierachical trees with k-means clustering



(d) hierachical trees with k-means clustering

Figure 8: Search precision vs. speed (dashed lines) for 9 different trees (number of features for all plots: 9000). Angle noise was set to $\sigma_{\text{angle}} = 1.5°$, position noise to $\sigma_{\text{pos}} = 0.05\text{m}$. The accuracy is related to the precision at this particular noise level for the linear search. Parameters: k-means *branching factor* and *number of iterations*, kd-tree *number of trees*.

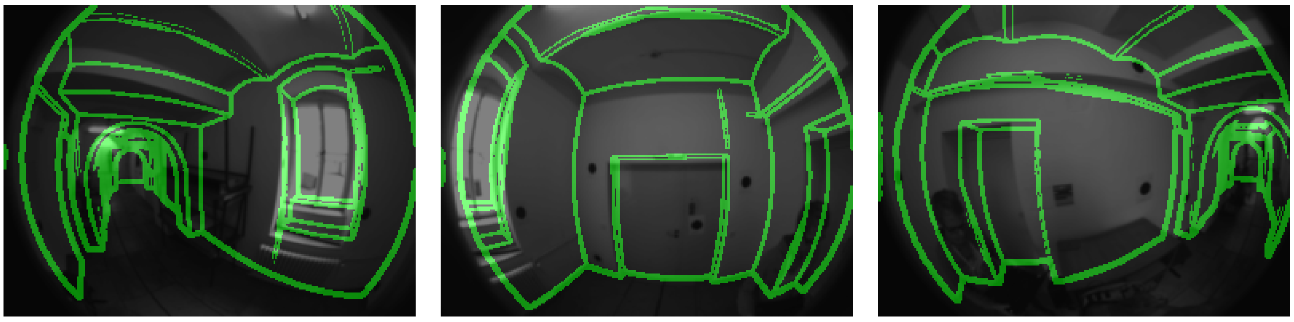


Figure 9: All three fisheye images overlaid with model edges after correct self-localization with the real image sequence. Complete sequence see `http://www2.ipf.kit.edu/Projekte/3DTracks/`.

Micušík, B., 2004. Two-view geometry of omnidirectional cameras. PhD thesis, Czech Technical University.

Micušík, B. and Pajdla, T., 2006. Structure from motion with wide circular field of view cameras. Pattern Analysis and Machine Intelligence, IEEE Transactions on 28(7), pp. 1135–1149.

Muja, M. and Lowe, D. G., 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In: International Conference on Computer Vision Theory and Application (VISSAPP'09), INSTICC Press, pp. 331–340.

Ottlik, A. and Nagel, H.-H., 2007. Initialization of model-based vehicle tracking in video sequences of inner-city intersections. International Journal of Computer Vision 80(2), pp. 211–225.

Puig, L., Bermúdez, J., Sturm, P. and Guerrero, J., 2012. Calibration of omnidirectional cameras in practice: A comparison of methods. Computer Vision and Image Understanding 116(1), pp. 120 – 137.

Reitmayr, G. and Drummond, T., 2006. Going out: Robust model-based tracking for outdoor augmented reality. In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR).

Scaramuzza, D., Martinelli, A. and Siegwart, R., 2006. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In: Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on, IEEE, pp. 45–45.

Simon, G., 2011. Tracking-by-synthesis using point features and pyramidal blurring. In: International Symposium on Mixed and Augmented Reality (ISMAR), pp. 85–92.

Steger, C., 2002. Occlusion, clutter, and illumination invariant object recognition. International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences 34(3/A), pp. 345–350.

Svoboda, T., 1998. Motion estimation using central panoramic cameras.

Ulrich, M., Wiedemann, C. and Steger, C., 2009. Cad-based recognition of 3d objects in monocular images. In: IEEE International Conference on Robotics and Automation, Kobe, Japan, pp. 1191–1198.

Weinmann, M., Wursthorn, S. and Jutzi, B., 2011. Semi-automatic image-based co-registration of range imaging data with different characteristics. In: PIA11 - Photogrammetric Image Analysis, Vol. 38 Part 3/W22, pp. 119–124.

Wuest, H., Vial, F. and Stricker, D., 2005. Adaptive line tracking with multiple hypotheses for augmented reality. In: Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR).

Wuest, H., Wientapper, F. and Stricker, D., 2007. Adaptable model-based tracking using analysis-by-synthesis techniques. In: 12th International Conference on Computer Analysis of Images and Patterns (CAIP), LNCS, Vol. 4673, Springer Berlin/Heidelberg, Wien, pp. 20–27.