Wide Area Localization on Mobile Phones

Clemens Arth, Daniel Wagner, Manfred Klopschitz, Arnold Irschara, Dieter Schmalstieg*
Graz University of Technology , Austria

ABSTRACT

We present a fast and memory efficient method for localizing a mobile user's 6DOF pose from a single camera image. Our approach registers a view with respect to a sparse 3D point reconstruction. The 3D point dataset is partitioned into pieces based on visibility constraints and occlusion culling, making it scalable and efficient to handle. Starting with a coarse guess, our system only considers features that can be seen from the user's position. Our method is resource efficient, usually requiring only a few megabytes of memory, thereby making it feasible to run on low-end devices such as mobile phones. At the same time it is fast enough to give instant results on this device class.

Index Terms: I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—3D/stereo scene analysis I.4.8 [Image Processing And Computer Vision]: Scene Analysis—Tracking; I.5.4 [Pattern Recognition]: Applications—Computer Vision C.5.3 [Computer System Implementation]: Microcomputers—Portable devices (e.g., laptops, personal digital assistants)

1 INTRODUCTION

In this work we aim at localizing a mobile user's 6DOF pose directly on a mobile phone using the phone's built-in camera. It is therefore important to consider the unique limitations introduced by this platform. For our purposes, the most important platform considerations are computing speed, memory and storage size as well as camera capabilities.

All units in a mobile phone are primarily optimized for low power consumption rather than raw processing speed. Additionally, memory is slow and caches are small, such that cache misses create serious performance hits. Code that is well written for a modern mobile phone will still run about 20-40 times slower than on a modern PC.

Another important factor is memory size. Limitations in the mobile phone operating systems do usually not allow more than about 5-10 MB per application. However, modern smart phones possess storage capabilities of several gigabytes, making hierarchical data handling attractive. Although the file storage is too slow for live data processing, it can be used for out of core processing or to cache large datasets to prevent continuous downloading.

Finally, the mobile phone's camera contributes to the level of detection quality that can be expected. We have seen enormous improvements in the still image taking quality of mobile phone cameras. Multi-megapixel cameras are common today, and given enough light, the quality is now comparable to compact cameras. Video capabilities, however, have only minimally improved, making it hard to select a good device for Augmented Reality (AR) usage. A severe limitation comes from the small field of view of mobile phone cameras, which typically lies in the range of $50^\circ - 60^\circ$, much less than what is common used in computer vision research. The limitations outlined above require developers to build a solution from the ground up for mobile phone platforms.

*e-mail: {arth,daniel,klopschitz,irschara,schmalstieg}@icg.tugraz.at

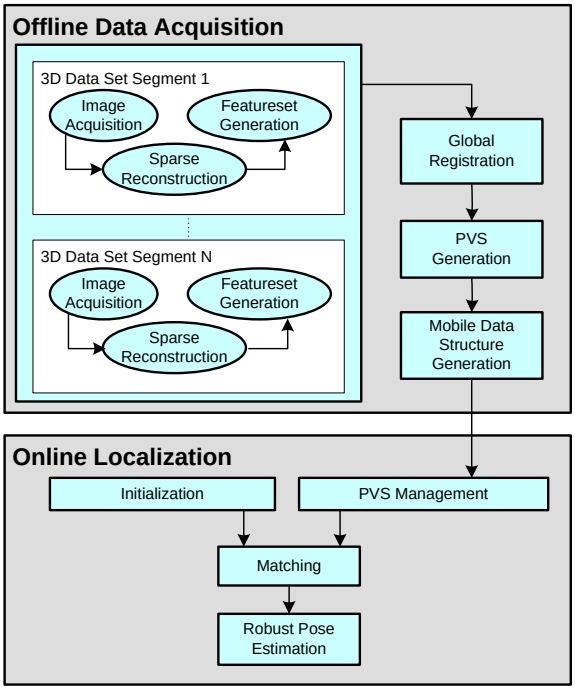

Figure 1: *Work flow overview.* The system can be divided into an offline data acquisition and an online localization part. The offline data acquisition step creates globally registered 3D points with associated feature descriptors that are grouped into potentially visible sets. Optimized representations of this data are used on the mobile phone to localize the user efficiently.

We propose a system for full 6DOF pose estimation from wide-area 3D reconstructions based on natural features, which is scalable and allows for highly efficient self-localization on mobile phones. To the best of our knowledge, there is no previous work in this research area that uses the idea of sparse 3D reconstructions and visibility information.

Our approach builds upon the idea of using image based reconstruction methods from computer vision to solve the data acquisition step. Because the image based reconstruction methods operate on the same type of data as the image based localization and pose estimation, it is straightforward to feed data from a 3D reconstruction into a localization system. Furthermore, reconstruction and localization both depend on texture information present in an environment. This means that if an environment is suitable for image based localization, it is usually also suitable for image based reconstruction and vice versa. A major advantage of this approach is that we do not have to segment our environment into explicit tracking targets and we are not limited to planar or other special geometrical structure. On this account we propose to use a globally registered image based reconstruction of an environment as a large 3D tracking target.

Because real world buildings usually contain untextured areas that are not suitable for image based reconstruction or localization

we use a pragmatic approach to create large-scale reconstructions. We divide the problem of reconstruction into suitable blocks and do not try to reconstruct complete buildings in one step. In this work, we manually register individual reconstruction into a single global coordinate system to create a single large globally registered 3D reconstruction which serves as a base for self-localization. Automatic global registration of reconstructed parts is left for future work.

Our goal is to build a system which is highly efficient and can be run on mobile phones. In order to accomplish this, the reconstruction has to be represented efficiently in terms of memory consumption, scalability and computational handling. Apart from that, a pose estimation algorithm must be fast, yet robust to deliver stable results even under the presence of a significant amount of noise and outliers. We propose an approach for generating a compact representation of 3D reconstructions in an indoor environment which is highly efficient in terms of memory consumption and handling on mobile phones.

Taking advantage of well-known approaches from computer graphics for visibility estimation, we propose a method for dividing large 3D reconstructions into several smaller parts that can be compactly represented and allow for building a highly scalable system. We build upon the idea of potentially visible sets (PVS) originating from work in the computer graphics community. Although PVS have a long tradition, to the best of our knowledge there exists no related work about their use in tackling computer vision problems.

We start the presentation with reviewing related work in the area of AR and robust localization in Section 2 . The system proposed in this paper can be divided into two major parts (see Figure 1). The first part is described in Section 3. It deals with the offline creation of the image based 3D reconstructions and the generation of highly efficient database structure. Furthermore, we describe how to generate a special cells-and-portals representation, which can be handled efficiently on mobile phones. The second part is presented in Section 4. It describes the online process of 6DOF pose estimation, *i.e*., the task of self localization within an environment represented as sparse 3D reconstruction of natural features. This process covers feature calculation, database management and robust pose estimation on a mobile phone. Experimental results are delineated in Section 5. The paper concludes with a summary and an outlook on future work in Section 6.

2 Related Work

In the following we review related work in the area of localization for AR, as well as previous work on localization purely based on computer vision techniques. Furthermore we introduce some work previously done for location recognition on mobile phones and the occlusion handling principle used in our approach that originates from well established concepts in computer graphics.

2.1 Localization for Augmented Reality

Among the first dedicated wearable location systems was the Active Badge system [36], which consisted of infrared (IR) badges sending location information signals to a server. Its successor, the Bat system [1], used ultrasonic location estimation to provide more accurate position data. PlaceLab [24] is a system that relies on signal strength of existing infrastructure, such as GSM, Bluetooth and WiFi, for indoor and outdoor location tracking. Accuracy strongly depends on the number of senders in the environment and has been reported in the range of 3-6 meters for indoor usage.

Markers have a strong tradition in Augmented Reality due to their robustness and low computational requirements. Hence, marker based approaches have also been applied for indoor localization. Kalkusch *et al*. [11] employed localization by tracking 2D barcodes (fiducial markers) that were installed in the environment. The IS-1200 tracker [20] is a commercial variant of this approach, combining tracking of circular fiducials with an inertial sensor for increased robustness. Although there is a large body of publications for indoor and outdoor tracking for Augmented Reality, little work has gone into localization. Reitmayr proposed an accurate localization technique [25] based on modeling the GPS error with a Gaussian process for fast outdoor localization without user intervention.

Markerless registration for Augmented Reality on mobile phones has only recently become possible due to increased processing capabilities of modern smart phones. First approaches were based on optical flow based methods such as TinyMotion [35]. Our own previous work [33] marks the first real-time 6DOF pose estimation from natural features on a mobile phone. However, none of these works addresses wide area tracking.

2.2 Vision based localization

In the computer vision literature, the problem of location recognition has been addressed in the past by a variety of approaches [26, 38, 39]. The most successful methods rely on wide baseline matching techniques based on sparse features such as scale invariant interest points and local image descriptors. The basic idea behind these methods is to compute the position of a query image with respect to a database of registered reference images [27], planar surfaces [26] or 3D models [10, 21]. Assuming a static scene, geometric verification can be used to determine the actual pose of the camera with respect to the exemplar database. Different viewpoints or illumination changes are largely handled by robust features like SIFT [16] and SURF [3] that act as descriptors of local image patches. These features have been found to be highly distinctive and repeatable in performance evaluation [18].

For example, Schindler *et al*. [27] present a city scale location recognition approach based on geo-tagged video streams and specific trained vocabulary trees using SIFT features. The vocabulary tree concept and inverted file scoring as described in [23] allows sub-linear search of large descriptor databases requiring low storage space. In contrast, Lepetit *et al*. [14] recast matching as a classification problem using a decision tree and trade increased memory usage for expensive computation of descriptors at runtime.

Skrypnyk and Lowe [28] present one of the first systems for 3D scene modelling, recognition and tracking with invariant image features. First, a sparse 3D model from the object of interest is reconstructed using multi-view vision methods. Second, SIFT descriptors associated with the sparse 3D points are organized into a kd-tree structure, and a best-bin first search strategy is employed to establish putative 2D-3D correspondences. A robust pose estimation algorithm is used for geometric verification and delivers the accurate pose of the query image with respect to the 3D model.

Based on the same fundamental concept, Irschara *et al*. [10] present a location recognition system for large scale scenes with full 6DOF localization. A precomputed set of synthetic views provides 3D point fragments that cover the space of admissible viewpoints. The 3D point fragments are globally indexed with a vocabulary tree data structure that is used for coarse matching. Real time performance for view registration on a desktop PC is achieved by utilizing modern graphics processing units for feature extraction and matching.

2.3 Localization for mobile phones

Because phones did not have sufficient computational power, the first approaches for mobile phone localization were based on a client-server architecture [5, 8]. To overcome resource constraints of mobile phones, tracking is outsourced to a PC connected via a wireless connection. All of these approaches suffer from low performance due to restricted bandwidth as well as the imposed infrastructure dependency, which limits scalability in the number of client devices. The AR-PDA project [6] used digital image streaming from and to an application server, outsourcing all processing

tasks of the AR application reducing the client device to a pure display plus camera. Hile reports a SIFT based indoor navigation system [8], which relies on a server to do all computer vision work. The server-based approaches are not suitable for AR; typical response times are reported to be about 10 seconds for processing a single frame.

However, recent approaches have shown that natural feature tracking with 6DOF can be done in real-time on mobile phones [33]. Takacs *et al*. [31] present an outdoor localization system working on mobile devices. Keypoint detection and matching is performed directly on the mobile phone. Features are clustered in a 2D grid and pre-fetched by proximity. Each grid element contains a set of clustered meta-features that represent the most repeatable and stable features of the scene. Geometric consistent meta-features are created. However, in contrast to our approach no 3D model is reconstructed and thus true geometric consistency is not enforced and no full 6DOF pose is computed.

2.4 Potentially visible sets

In computer graphics and virtual reality applications, visibility is a fundamental problem, since it is necessary for occlusion culling, shadow generation and image based rendering. One of the earliest methods addressing the problem of visibility culling is the potentially visible set (PVS) approach [2]. The basic idea is to discretize the environment into view cells and precompute the cell-to-cell visibility. In densely occluded environments, such as hilly regions, urban areas or building interiors, the potentially visible sets significantly reduces the amount of data that has to be processed for rendering. Indoors, the natural structure of cells (rooms) and portals (doorways) can be easily exploited [32]. In our localization system, we take advantage of the PVS data management by splitting the 3D model into chunks of 3D points that are organized by visibility constraints. Thus, the potentially visible set determines the number of 3D points and descriptors that have to be considered according to the current view cell. This is in contrast to the 2D grid method of Takacs *et al*. [31], which uses a regular subdivision on a map to partition the data and does not exploit visibility.

3 Offline data acquisition

This section explains the steps involved in creating the 3D tracking data in more detail and motivates some design decisions. Triangulated natural image features, obtained with a structure from motion (SfM) system, are used as a basis. These 3D points are registered into a global coordinate system and partitioned into a representation suitable for efficient localization on mobile phones. We use an offline data acquisition step instead of building the tracking data on-the-fly like Simultaneous Localization and Mapping (SLAM). Recent work on SLAM such [12] can efficiently map small workspaces, but has considerable computational cost and does not consider global registration or sharing of reconstructions, which is implicitly addressed in our work.

3.1 Structure from motion

Current SfM methods may be classified based on the image data they use. Two major types of image sources are still images and video sequences. Recent trends in photo community websites and the ubiquitous availability of digital cameras have shifted the research interest towards reconstruction from unordered image data sets [30]. This choice of input data usually has strong implications on the selection of algorithms used to solve the SfM problem. Unordered sets of still images rely on wide baseline image matching techniques to establish correspondence information, and the SfM problem may be solved for all input images simultaneously or in an incremental way [9, 17, 29].

Video stream based image capturing allows for simpler feature point tracking, but cannot be used effectively in narrow and cluttered indoor environments. The movement of the observer is usually constrained. We use sets of unordered still images obtained from suitable points in the environment, and a well behaved trajectory between these points is therefore not necessary. The disadvantage of this approach is that the feature matching process is more challenging than feature tracking in a video. However, in our experience a still image based method is more flexible for 3D reconstruction. Furthermore, our generic approach allows to incrementally extend the 3D models with new images over time.

Image acquisition

We divide the problem of collecting source images for the sparse natural feature reconstruction into suitable blocks. We do not try to reconstruct a complete building in one step. The reason for this strategy is that texture is necessary for image based reconstruction and of course later for the natural feature based localization. Real world buildings usually contain untextured areas, which are not suitable for image based reconstruction or localization. However, floor plans of buildings are usually available and allow for fast manual global registration of smaller reconstructions. It turned out that a room sized granularity of individual reconstructions works well in practice and also allows the replacement of a reconstruction with a newer one if the appearance of a room changes significantly.

We use a digital SLR camera with a large field of view ($\alpha = 90^\circ$) and high resolution (8 megapixel) for indoor image data acquisition. This combination reduces the number of images that are necessary to cover a room and makes the image to image correspondence computations needed for SfM more robust, because weakly textured areas are less likely to cover large portions of an image. Radial lens distortion is usually larger for short focal length optics and has to be removed from the images. We use a pre-calibrated camera with known distortion for SfM.

Reconstruction

The SfM problem is solved for each reconstruction segment separately. Given a set of unordered input images, SIFT features [16] are extracted from every image. A vocabulary tree [23] is used for coarse matching of similar images. This greatly reduces the computational effort of pair-wise image matching, by only matching the most relevant images as reported by the vocabulary scoring. The vocabulary tree is trained in an unsupervised manner with a subset of $2,000,000$ SIFT feature vectors randomly taken from 2500 images. Thus the vocabulary is generic and can be generalized to handle different data sets. The descriptor vectors are hierarchically quantized into clusters using a k-means algorithm.

The tentative sparse image correspondences retrieved from the vocabulary tree are then matched using an approximated nearest neighbor technique. The epipolar geometry is computed using a five-point [22] minimal solution to the calibrated relative pose problem inside a RANSAC [4] loop. The scale between the different epipolar geometries is estimated and the local geometries are merged into one coordinate system in a robust way. We rely on previous work to carry out this step [9, 13, 37].

The number of images in a reconstruction segment is typically between 50 and 300. Figure 2 shows an example of a room that was reconstructed using our approach.

Feature extraction and triangulation

While the reconstruction pipeline uses a standard SIFT key point detector and descriptor, the detection on the mobile phone must be optimized for speed and therefore uses a proprietary method (Section 4.1). Consequently, we must extract these proprietary features from the original images and triangulate them using the camera poses reconstructed in the SfM step.

This adds no significant overhead to the reconstruction process. because the computational effort of pair-wise image matching has

(a)

(b)

Figure 2: *3D segment.* Figure (a) shows some of the input images used to create an initial sparse 3D reconstruction. Figure (b) shows two views of this reconstruction, triangulated natural feature points are shown as dark points, the camera positions and orientations of the input images are visualized by the coloured cones.

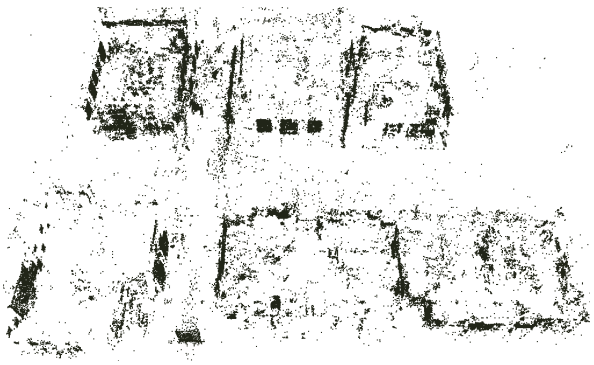

Figure 3: Global registration of individual reconstruction into a common coordinate system.

already been solved during reconstruction and well matching image pairs are already known. Unfortunately, the extracted features are not free of outliers originating from wrong image matches. To remove most wrong features, we filter them by enforcing additional criteria: A feature visible in two views must also be visible in a third view, and the reprojection error must be smaller than a predefined threshold. This method yields an almost outlier-free set of new features.

3.2 Global registration

Our goal is to compute the full 6DOF pose, therefore each 3D reconstruction has to be registered relative to a global coordinate system. We find this similarity transform by aligning each of our newly created reconstructions manually to a 2D floor plan of the building. All 3D points used for localization are transformed into a single global coordinate system. Another option would be to compute the 6DOF pose relative to a reconstruction segment and transform the result into the global coordinate system. The advantage of transforming all 3D points into one coordinate system is that 3D points from multiple reconstruction segments can be used simultaneously during the localization pose computation. This may happen if feature points from multiple reconstruction segments are visible in the same image. Figure 3 shows an example of eight globally registered reconstruction segments.

3.3 Potentially visible sets

Since the overall feature database does not fit into memory, it is necessary to split the dataset into chunks that can be loaded independently. Takacs *et al.* [31] suggested to partition the database into cells using a 2D regular grid. In their approach, the computer always holds the feature sets of the closest 3x3 cells in memory. In contrast, we split the dataset by visibility, using a PVS structure.

We follow a common practice that PVS are often built at the same granularity as the cells that they index: For every cell there exists one PVS that refers to all other cells that are visible from inside this cell (see Figure 4). Each cell stores a list of all features that are located inside its area. The associated data is stored in a separate file and loaded on demand. Figure 5 shows such a partitioning of a larger 3D reconstruction into individual cells. While automated techniques to compute the PVS exist [32], for simplicity the PVS structure used in our evaluations was created by hand.

For each feature, we also store the indices of all images containing the feature. This additional information can be used to vote for areas that are likely to be seen in a camera image. For each PVS, we build a k-means tree for fast searching and voting, using Lloyd's algorithm [15]. Additionally, we built an inverse file structure in the form of additional k-means trees for all original camera images that where used to reconstruct the PVS. The inverse file structure allows matching against only those features that come from a specific image. This approach greatly improves the matching rates, but comes at the price of increased memory usage for the search structures and probably a reduced matching accuracy.

We experimented with various branching factors to build the k-means trees and found a branching factor of 10 to deliver good results at reasonable speed. In our tests, a larger branching factor did not deliver significantly better results. We grow the trees until a given maximum number of descriptors fits into the leaf nodes. In our current implementation we target an average of 45 descriptors

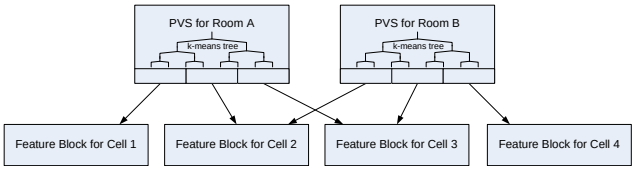

Figure 4: Two PVS with three cells each, sharing two of the cells.

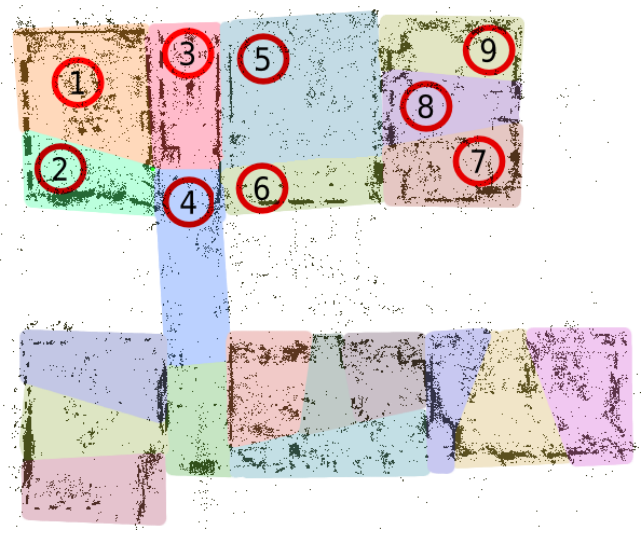

Figure 5: Representation of the reconstruction as separate PVS cells. The assignment of features to cells is color coded. The first 9 cells are indicated by the numbers inside red circles. Because we use cell based visibility, a PVS is created for each of these color coded cells and consists of an enumeration of all cells that are visible from one cell. For example, the PVS of cell 2 consists of cells $(2,1,4)$.

per leaf node. When searching for a match, we traverse the tree until encountering a leaf node and then compare against all descriptors in that node. Hence, we do not require back tracking.

4 Online Localization

In the following we describe the online workflow on the mobile phone to estimate a valid 6DOF pose on a mobile phone.

4.1 Fast and robust descriptors

In previous work [33], we developed an extremely fast descriptor called PhonySIFT. It is loosely based on SIFT, but designed to operate in real time on a mobile phone. One restriction is that tracked objects must be planar, which allows highly effective outlier removal, yielding enough robustness for typical AR applications. However, for the use case targeted in this paper, the PhonySIFT descriptor is not sufficient. The outlier removal techniques used in [33] do not work for 3D objects, and more importantly its robustness does not scale well enough to match against tens of thousands of key points.

On this account, we developed a new proprietary detector/descriptor combination which was inspired by SURF [3]. Key points are identified in scale space and a histogram of gradients is built. Tests done with the framework of Mikolajczyk *et al.* [18, 19] show that it performs as well as SIFT and SURF and sometimes outperforms both. However, we noticed that SIFT performs better in some real-world scenarios, which is why we continue to use it for the 3D reconstruction. Our detector/descriptor is several times faster than the publicly available implementation of SURF[1] and faster than a GPU-SIFT implementation (comparing our method on a single-core of a 2.5GHz Intel Core 2 Quad against GPU-SIFT on an NVIDIA GeForce GTX 280). For localizing and describing keypoints in a 640x480 image, our descriptor requires about 20ms on that 2.5GHz CPU and about 120ms on a mobile phone[2].

4.2 Run-time memory management

Feature and PVS data is stored in a format that is optimized for fast loading, which includes storing items in large chunks and minimizing the number of memory allocations required. The feature datasets of the cells are stored in feature blocks. A custom memory manager loads and discards feature blocks by counting references from the PVS structures to the cells. When a cell is no longer required by any PVS, the memory manager can discard it to free memory for other cells. Feature blocks are loaded on demand when a PVS requests them. In our tests, a feature block typically has a memory footprint of ∼1-2MB and a PVS is built from 2-4 cells. The memory footprint of the PVS is dominated by the search structures and range between ∼1-2MB. The overall memory footprint is ∼5MB, which is small enough to fit into a mobile phone's application memory.

4.3 PVS selection

Since only a part of the whole dataset is in memory at a given time, no global localization can be performed. Consequently, it is necessary to initialize the localization process by giving the system a hint about where to search. In the following, we outline several variants for providing an initial location for the PVS.

The most user friendly approach relies on additional sensors and therefore does not require any user interaction. Outdoors, GPS can give a sufficiently accurate position for searching a single PVS only. Indoors, WiFi triangulation, Bluetooth or infrared beacons are able to deliver a coarse location, which requires searching the closest few cells. If no sensor data for automatic coarse localization is available, the user can select the current position on a map or from a list (rooms, streets, etc). If the tracked object or environment does not have a unique location (for example, a consumer product for advertising), the dataset can be manually selected or determined using a barcode on the object. Once the mobile device is localized, the system can switch into an incremental tracking mode.

Compared to first initialization, a re-initialization step can then benefit from the fact that a previous position is known. If the time between losing tracking and re-initialization is short, it makes sense to restrict the search area to the PVS of the last known position. However, in many practical scenarios tracking interruptions will last longer (*e.g.*, many users may put a handheld device down while walking). Depending on the assumed speed of a user, the search area must therefore be widened over time until it reaches a level that can not be searched in a meaningful time anymore.

4.4 Localization

Localizing a mobile users position involves the following steps: feature extraction and description, feature matching, optional voting, outlier removal, and finally pose estimation and refinement. Feature extraction uses a scale space search to find keypoints in the 2D image, including a size estimation. For each keypoint, we estimate a single dominant orientation and create one descriptor. The scale space search step dominates the resource requirements, taking about 80% of the computation time and requires roughly 12 bytes per camera image pixel. The memory overhead for creating descriptors is relatively low with about 0.3 bytes per camera image pixel and ∼80 bytes per feature.

[1] http://code.google.com/p/opensurf1

[2] calculated on an *ASUS P565* mobile phone with an Intel XScale processor running at 800MHz for about 240 features.

We implemented two alternative methods for feature matching: in matching-friendly scenarios, we directly match all camera image features against all features in the current PVS. Alternatively, we use a vocabulary tree voting scheme: We first define subsets by finding those images from the reconstruction step that contain enough features that match the current camera image. We then match against the top ranked subsets separately. For each subset we then try to estimate a pose. The advantage of this two step approach is that we largely reduce the number of features to match against, which makes the matching itself more robust. However, it has higher computational requirements.

In both cases, matching the camera image against the dataset gives a set of 2D-3D correspondences that still includes outliers. A robust pose estimation procedure is therefore required to deal with these outliers. We therefore apply a RANSAC scheme with a 3-point pose [7] as hypothesis and use a subset of up to 50 correspondences for validation. The 3-point pose estimation is based on a fixed-point implementation of the method in [4].

The hypothesis with the largest number of inliers is selected as a starting point for a non-linear refinement. Based on the inlier set of the best hypothesis, we apply an M-estimator in a Gauss-Newton iteration to refine the pose and find more inliers. This step is repeated until the inlier set does not grow anymore. In theory, four points are enough to calculate a 6DOF pose from known 2D-3D correspondences. However, given a large enough number of outliers, it is likely to find an invalid pose from a small number of correspondences only. We therefore treat a pose only as valid if at least 20 inliers were found.

4.5 Detection vs. tracking

Tracking by detection at every frame is common, because detection is always required and tracking is then solved at the same time. Our wide-area detection system runs in about 1/3 of a second on a fast mobile phone and about 10 times faster on an average PC. It would therefore, at least on the PC, qualify for tracking by detection.

However, tracking by detection has several disadvantages compared to an approach with separate detection and tracking stages as in [34]. A pure detection system does not take advantage of frame-to-frame coherence and therefore always searches without a strong prior. This requires more computational effort and reduces the chances of finding a correct solution. A tracker pre-selects the features to look for using a motion model and therefore reduces the search space. It is therefore typically more robust and efficient than a detector. However, tracking usually requires a different kind of dataset compared to the detector, which means that two separate datasets have to be created, maintained, stored and held in memory.

5 Experiments

This section presents evaluation results for the wide-area localization. For simplicity, the test cases are based on indoor reconstructions taken at the university campus. Note that although we do not explicitly provide results on outdoor scenarios, our methods are also fully applicable outdoors.

5.1 Test data acquisition

For testing our approach we generated a fully registered 3D reconstruction of several adjacent rooms in an office building on the university campus, using a digital SLR camera and the methods described in Section 3. For the localization experiments, two smartphones were used, a *Meizu M8*[3] and an *Asus P565*. One of them, the *M8*, was used to take a set of 94 high resolution pictures while walking along a specified path through three rooms of this reconstruction. For every position, two images were taken, one with the built-in lens of the *M8* and one with a special adhesive wide-angle

[3] http://www.meizu.com

Figure 6: Sample images from the test set collected along a path through three rooms of our department. On the left side, images taken with the normal lens are shown, on the right side pictures taken from the same position with the wide-angle lens are depicted.

Figure 7: *Meizu M8* mobile phone front view, back view with magnetic mounting ring, wide angle camera lens from *AmacroX*, mounted camera lens (from upper left to lower right).

Database	PVS 1 (cells 1,2,4)	PVS 2 (cells 2-4,6)	PVS 3 (cells 4-6)	Grid (whole area in Fig. 3)
Size	5,378 kB	5,159 kB	2,348 kB	16,789 kB

Table 1: Amount of memory needed for storing the individual working sets, the PVS as marked in Figure 5 and the entire reconstruction as shown in Figure 3.

lens from *AmacroX*[4], which can be mounted on a phone. A collection of images for both lens types is shown in Figure 6. The phone was mounted on a tripod to ensure that the image pairs with and without the *AmacroX* lens were taken from the exact same position. The mobile phone and the setup are depicted in Figure 7. For simplicity, we calibrated the camera once with and without the *AmacroX* lens, but did not recalibrate the camera each time after reattaching the wide angle lens.

Using two different lenses makes the optimal choice of parameter settings for the various algorithms more difficult. Moreover, effects resulting from the radial distortion of the *AmacroX* are difficult to analyze. Nevertheless, we included the image sets obtained with both lenses to better understand the effect of field of view in complex localization problems.

5.2 Memory consumption

We investigated the effect of using a PVS structure on the size of working set selected from the overall database. The memory consumption of the PVS structure is compared to a regular subdivision as used by by Takacs *et al*. [31]. Results for the working set consisting of descriptors k-means tree, using 8-bit quantized values for every histogram entry, are listed in Table 1.

Note that the amount of memory for the databases in the individual PVS is considerably smaller than the amount needed for an entire area. Also note that due to our management of sharing cells among PVS structures (see Section 4.2), the amount of memory is even smaller. This means that the values in Table 1 refer to the maximum amount of memory needed and include shared cells.

5.3 Matching strategies

In the preparation of our performance tests, we evaluated different strategies for generating correspondences between features detected in a given 768x576 pixel image and the features contained in a database (refer to Table 2 for averaged results). We tested exhaustive matching (1) and k-means tree based matching with subsequent histogram filtering, where the best 3 candidates were evaluated. For matching with the features of the best candidates we tested two configurations, one using exhaustive matching (2) and one using a k-means tree (3). For all three strategies we matched five test images against a database containing 40,670 features.

In the query images, on average 784 features were detected. The runtimes for the second and the third approach also include the time needed for histogram voting. Note that the second and the third method only approximate the result, while only the first method delivers the exact matching. Nevertheless it is easy to see that in terms of the computation time, the third strategy outperforms both other methods by far. Consequently is was adopted as our preferred method for our online localization For a comparison of the accuracy, the reader is referred to the next experiment.

5.4 Image resolution

The aim of this experiment is to evaluate the influence of image resolution on robustness. Starting with an initial resolution of 1920x1440 pixels (the maximum still image resolution of the *Meizu M8* camera), we iteratively reduced the image resolution down to

[4]http://www.amacrox.com

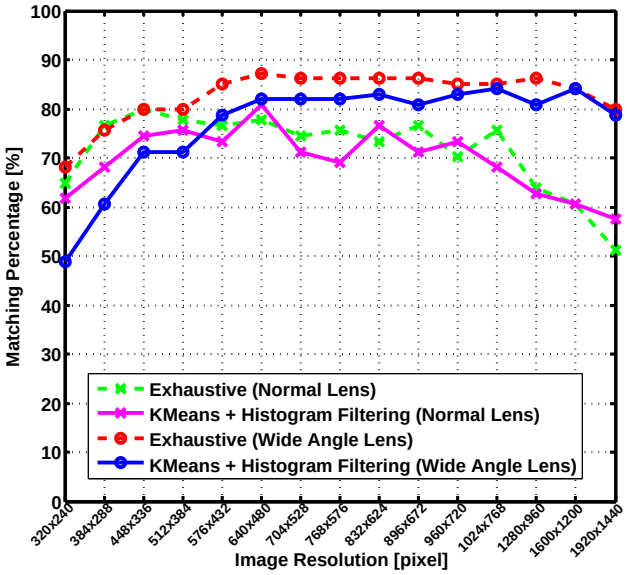

Figure 8: Image resolution vs. matching percentage for both lens types and both matching methods. The results for the wide-angle lens are drawn with dashed lines.

Strategy	Exhaustive Matching (1)	Tree-based Matching (best 3 candidates exhaustively) (2)	Tree-based Matching (best 3 candidates tree-based) (3)
Time	17.95 s	1.660 s	150.78 ms

Table 2: Runtime performance for different matching strategies on the *ASUS P565*.

320x240 pixels. For all image resolutions, we allowed a reprojection error of 1.5 % of the corresponding focal length, which is equal to about 20 pixels in the largest resolution. In the lowest image resolution, this is equivalent to about 3.3 pixels. To keep the rate of features detected in the images at a reasonable level, we lowered the threshold for detecting features with decreasing image resolution, starting with a relatively high threshold. This was mainly done to remove the large amount of noise due to very small features detected in larger image resolutions.

To prove the performance of our matching approach based on individual k-means trees and histogram-based filtering of candidates (with consecutive k-means tree based matching), we compared it with an exhaustive search over the whole database, *i.e*., all feature points contained in the reconstruction. The results of this evaluation are shown in Figure 8. Note that for convenience, this test was performed on a standard desktop computer.

It is easy to see that the matching performance for the image sequence acquired with the wide-angle lens is generally better than for the sequence acquired with the normal camera lens, except for very low resolution. The advantage of using the wide-angle lens clearly stands out and highlights the importance of a wide field of view for localization.

Note that the matching performance does not increase for resolutions above 640x480 pixels, and may even slightly drop for higher resolutions. Using a 3D reconstruction procedure, such as the one described previously in Section 3, the resulting database contains image features which constitute a coarse to moderately detailed representation of the environment. Image features which represent fine details of objects (*e.g*., text on a wall poster) are usually not represented in the reconstruction. If high image resolutions are used for comparison, fine details may be extracted and can lead to increased

	Image Resolution	512x384	640x480	768x576	1024x768
	Avg. Num. of Features	372	557	784	1253
Meizu M8	Feature Extraction	278.30 ms	387.58 ms	593.11 ms	2,155.1 ms
	Histogram Voting	32.48 ms	48.18 ms	67.25 ms	109.96 ms
	Matching best (max.) 3 cand. + RP	129.15 ms	178.95 ms	214.24 ms	302.77 ms
	Total Time	**439.94 ms**	**614.71 ms**	**874.60 ms**	**2,567.83 ms**
ASUS P565	Feature Extraction	125.72 ms	193.19 ms	306.11 ms	481.85 ms
	Histogram Voting	26.69 ms	39.5 ms	54.59 ms	90.57 ms
	Matching best (max.) 3 cand. + RP	96.18 ms	120.20 ms	141.77 ms	237.79 ms
	Total Time	**248.59 ms**	**352.89 ms**	**502.43 ms**	**810.21 ms**

Table 3: Runtime performance for different image resolutions and the individual algorithms contained in our approach.

Individual Algorithm	Percentage [512x384]	Percentage [640x480]	Percentage [768x576]	Percentage [1024x768]
Feature Extraction	50.57 %	54.75 %	60.93 %	59.47 %
Histogram Voting	10.74 %	11.19 %	10.86 %	11.18 %
Matching best 3 cands.	21.61%	20.18 %	19.15 %	20.62 %
Robust Pose Estim.	17.08 %	13.88 %	9.06 %	8.73 %

Table 4: Average percentage of computational time spent in the individual parts of our algorithm on the *P565*.

mismatches. We partially overcome this by choosing a corresponding threshold for higher resolutions, but the effect is still noticeable. Another observable fact is that our preferred method of tree-based matching has only a ∼5-10 percent lower performance rate due to the errors in the matching stage.

Although we chose our limit for feature correspondences initially accepted by the robust pose estimator quite high, Figure 9 shows that almost 80 % of all inliers used for calculating the final pose for a 768x576 pixel image have a reprojection error smaller than 4 pixels. Thus we assume the resulting pose to be quite accurate, despite alignment errors introduced during manual global registration of individual reconstructions, and despite errors resulting from the triangulation step in a typical small-baseline indoor environment.

5.5 Full system mobile phone evaluation

Finally we performed an evaluation of our localization method with the two smartphones, the *Meizu M8* (800MHz ARM11 CPU with FPU) and the *ASUS P565* (800MHz Intel XScale without FPU). On the *M8* we ran a version of our software using hardware floating point, whereas on the *P565* we ran a version using fixed-point math. We tested several different image resolutions for five test images, averaging the results. The database used is PVS 1 from our reconstruction, containing 40,670 features (equaling the same number of reconstructed 3D points). The results of our evaluation are listed in Table 3. A breakdown of the amount of computation time spent in the individual algorithms contained in our approach is given in Table 4.

As can be seen from row 2 of Table 3, the number of features increases almost linearly with the number of pixels of the query images and corresponds to the time spent in feature extraction. Interestingly, the CPU of the *M8* runs at the same rate as the CPU of the *P565*, but the performance of the *P565* is far better, which may

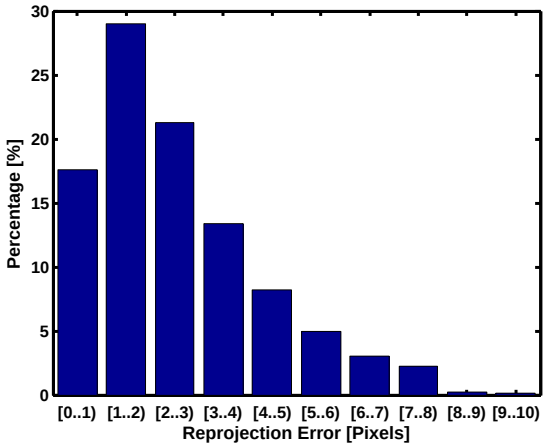

Figure 9: Reprojection error for inliers found during robust pose estimation. A reprojection error of ~ 10 pixels was allowed for an image size of 768x576 pixels.

be caused by caching or memory bandwidth limitations of the *M8*. Thus, for the rest of the discussion we focus on the results from the *P565*.

The amount of time spent in the histogram voting and the robust pose estimation stage only slightly increases, compared to the time spent in the feature extraction stage. The total amount of time for the whole process from feature extraction to robust pose estimation is about $248.59ms$ for a 512x384 pixel image, up to about $810.21ms$ for a 1024x768 pixel image. Comparing this result with the results from Figure 8, one can see that for a 768x576 pixel image a correct 6DOF pose (which is correct in about 82% of all cases) us computed at approximately $2fps$. Note that in this case the use of the wide-angle lens is assumed.

As can be seen in Table 4, the major amount of time is spent in the feature extraction stage. This is due to the computationally intensive task of scale-space search for extremal points and the subsequent generation of descriptors. The percentage of time spent on histogram voting for finding the best candidates and matching the descriptors stays almost constant across different image resolution. The time spent in the robust pose estimation stage drops due to the increased percentage of time spent in the feature detection stage. Moreover, our robust pose estimation method is able to discard a large number of outliers in a very early stage of the algorithm, thus keeping the overall amount of time needed relatively low (compare rows 5 and 9 in Table 3).

6 Conclusion and Outlook

In this paper we presented an approach for wide-area 6DOF pose estimation. It relies on a previously acquired 3D feature model, which can be generated from image collections, and can therefore tap into the rapidly increasing amount of real world imagery acquired for digital globe projects and similar ventures. To make the approach scalable, a representation inspired by potentially visible set techniques was adopted together with a feature representation that is suitable to work in real time on a mobile phone. Our evaluations show that robust recovery of full 6DOF pose can be obtained on a current smartphone at about 2-3Hz, which is sufficient to initialize incremental tracking methods.

Due to the complexity of the localization task and the large number of different algorithms involved, we cannot discuss all aspects of our system in detail. In this work, we focus on the general feasibility of computing a localization efficiently with the proposed methods. An evaluation of the behavior of localization performance

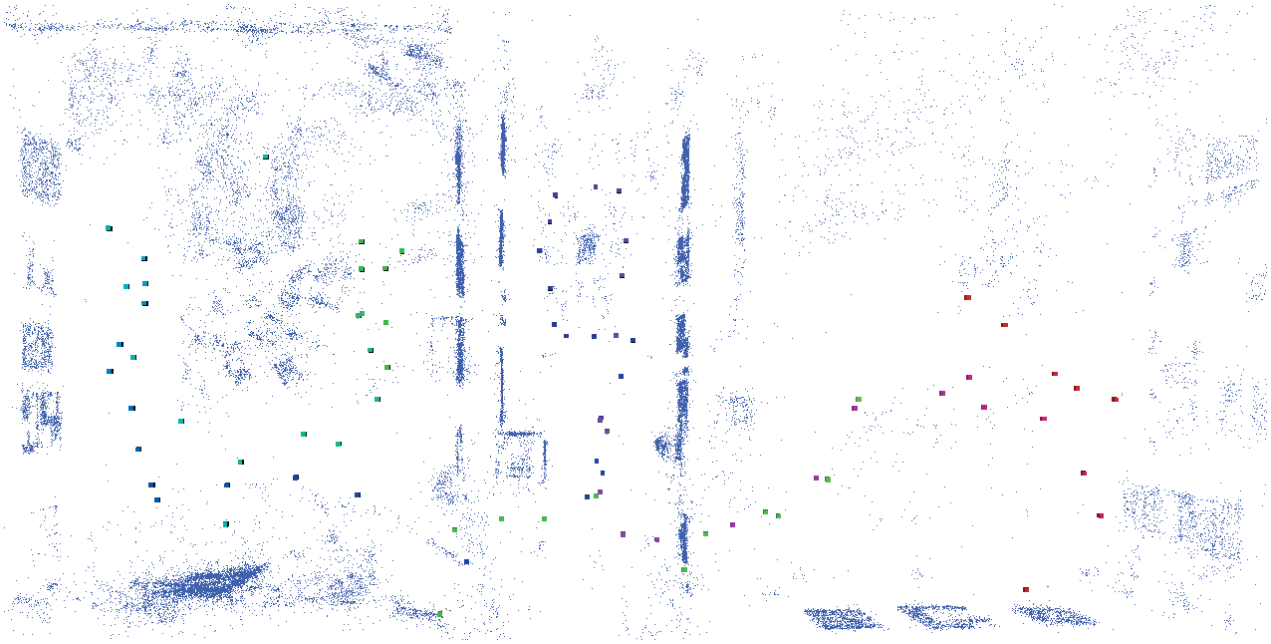

Figure 10: Three reconstructed rooms in which the set of test pictures was acquired. The camera poses estimated by our algorithm are represented as small, color-coded cubes from red via magenta and cyan to green. The path started in the right room (red), walking around in the middle room (magenta), going into the left room (cyan) and back into the right room (green).

over large time periods would be an interesting research topic and will be addressed in future work. From our experience, large parts of the individual reconstructions do not change much over time, and localization worked well when reconstruction data acquisition was carried out several weeks earlier. Nevertheless there are also parts of the reconstructions which are likely to change, and thus more advanced methods are needed to overcome the resulting problems. For example, a bookshelf seems to be a good tracking target, but the ordering of books on the shelf changes frequently, as users take books and put them back at different positions. Note that this also happened during our evaluations and caused some localizations to fail. This will be subject of further investigations concerning the frequent updating of existing reconstructions or invalidating outdated parts of reconstructions.

In the future, we plan to combine the localization with an incremental tracker, working from the pose calculated with the approach proposed here. However, probably different types of datasets are necessary to make this feasible [34]. In our current work, we did not include any investigations about special properties of cellular networks or wireless communication protocols necessary for online-sharing of 3D reconstructions from a centralized server. Nevertheless, this will be – together with a more in-depth investigation of PVS for efficient database management – subject to more closer research in the future as we plan to enlarge the reconstructed area at our department and think about deploying a test system for demonstration purposes.

Acknowledgements

This work was partially sponsored by the Christian Doppler Laboratory for Handheld Augmented Reality and the Austrian Science Foundation *FWF* under contract Y193 and W1209-N15.

References

[1] M. Addlesee, R. W. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, 2001.

[2] J. M. Airey, J. H. Rohlf, and F. P. Brooks, Jr. Towards image realism with interactive update rates in complex virtual building environments. In *Proc. Symposium on Interactive 3D Graphics*, pages 41–50, New York, NY, USA, 1990. ACM.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, June 2008.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[5] G. Fritz, C. Seifert, and L. Paletta. A mobile vision system for urban detection with informative local descriptors. page 30, 2006.

[6] J. Gausemeier, J. Fründ, C. Matysczok, B. Bruederlin, and D. Beier. Development of a real time image based object recognition method for mobile AR-devices. In *Afrigraph*, pages 133–139. ACM, 2003.

[7] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 592–598, 1991.

[8] H. Hile and G. Borriello. Information overlay for camera phones in indoor environments. In *Location- and Context-Awareness, Third International Symposium, LoCA*, volume 4718, pages 68–84. Springer, 2007.

[9] A. Irschara, C. Zach, and H. Bischof. Towards wiki-based dense city modeling. In *Workshop on Virtual Representations and Modeling of Large-scale environments (VRML)*, 2007.

[10] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[11] M. Kalkusch, T. Lidy, G. Reitmayr, H. Kaufmann, and D. Schmal-

stieg. Structured visual markers for indoor pathfinding, 2002.

[12] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, November 2007.

[13] M. Klopschitz and D. Schmalstieg. Automatic reconstruction of wide-area fiducial marker models. In *ISMAR*, pages 1–4, Washington, DC, USA, 2007. IEEE Computer Society.

[14] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages II: 775–781, 2005.

[15] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[16] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal on Computer Vision (IJCV)*, 60(2):91–110, 2004.

[17] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1615–1630, Oct. 2005.

[19] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *International Journal on Computer Vision (IJCV)*, 65(1/2):43–72, 2005.

[20] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE Computer Society, 2002.

[21] H. Najafi, Y. Genc, and N. Navab. Fusion of 3d and appearance models for fast object detection and pose estimation. In *Asian Conference on Computer Vision (ACCV)*, pages 415–426, 2006.

[22] D. Nistér. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–770, 2004.

[23] D. Nistér and H. Stewenius. Scalable recognition with a vocabulary tree. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2168, 2006.

[24] V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara. Accurate GSM indoor localization. In *Ubicomp*, pages 141–158, 2005.

[25] G. Reitmayr and T. W. Drummond. Initialisation for visual tracking in urban environments. In *ISMAR*, pages 1–9, Washington, DC, USA, 2007. IEEE Computer Society.

[26] D. Robertson and R. Cipolla. An image-based system for urban navigation. In *British Machine Vision Conference (BMVC)*, pages 819–828, 2004.

[27] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 0, pages 1–7, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[28] I. Skrypnyk and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 110–119, 2004.

[29] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *Proceedings of SIGGRAPH 2006*, pages 835–846, 2006.

[30] N. Snavely, S. M. Seitz, and R. S. Szeliski. Skeletal graphs for efficient structure from motion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[31] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Multimedia Information Retrieval*, pages 427–434, 2008.

[32] S. J. Teller and C. H. Séquin. Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Comput. Graph.*, 25(4):61–70, 1991.

[33] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, Cambridge, UK, Sept. 15–18 2008.

[34] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, Orlando, FL, USA, 2009. IEEE Computer Society.

[35] J. Wang, S. Zhai, and J. F. Canny. Camera phone based motion sensing: interaction techniques, applications and performance study. In *UIST*, pages 101–110. ACM, 2006.

[36] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, Jan. 1992.

[37] C. Zach, A. Irschara, and H. Bischof. What can missing correspondences tell us about 3D structure and motion? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[38] W. Zhang and J. Kosecka. Image based localization in urban environments. In *International Symposium on 3D Data Processing, Visualization and Transmission*, pages 33–40, Washington, DC, USA, 2006. IEEE Computer Society.

[39] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H. Sawhney. Real-time global localization with a pre-built visual landmark database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.