Wide-Area Scene Mapping for Mobile Visual Tracking

Jonathan Ventura*
University of California, Santa Barbara

Tobias Höllerer†
University of California, Santa Barbara

Abstract

We propose a system for easily preparing arbitrary wide-area environments for subsequent real-time tracking with a handheld device. Our system evaluation shows that minimal user effort is required to initialize a camera tracking session in an unprepared environment. We combine panoramas captured using a handheld omnidirectional camera from several viewpoints to create a point cloud model. After the offline modeling step, live camera pose tracking is initialized by feature point matching, and continuously updated by aligning the point cloud model to the camera image. Given a reconstruction made with less than five minutes of video, we achieve below 25 cm translational error and 0.5 degrees rotational error for over 80% of images tested. In contrast to camera-based simultaneous localization and mapping (SLAM) systems, our methods are suitable for handheld use in large outdoor spaces.

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

Accurate position and orientation information is crucial for wide-area augmented reality (AR), because displaying virtual content situated in the real environment requires knowledge of the exact pose of the display device. The task of autonomously determining a device's position and orientation with respect to a reference coordinate frame is called self-localization. Highly accurate self-localization enables sophisticated location-based applications, such as precise pinpointing of geographic features, accurately registered information overlays, and collaboration between multiple users in a shared space.

Todays mobile devices achieve self-localization by triangulation from satellites (GPS) and WiFi transmitters, along with a digital compass and inertial orientation sensors. Accuracy is limited with these approaches; for example, civilian GPS exhibits an accuracy of 7.8 meters at a 95% confidence level. The accuracy can be improved with augmentations such as differential GPS, although this is not typically available on consumer hardware. Another limitation with these sensors is the availability of a signal; often in urban environments, building walls interfere with satellite visibility, WiFi transmission, and magnetometer measurements.

A promising alternative is the use of computer vision techniques to recognize the visible scene and estimate the camera position and orientation. The camera pose can be estimated relative to a 3D model of the environment by matching features between the camera image and the pre-built model. Such techniques can provide a high level of positional accuracy, within 5-25 cm [26, 2]. However, there are two main challenges to implementation of wide-area visual localization: how to easily produce an environment model, and how to achieve real-time localization on a mobile device. Previous approaches typically use large-scale mapping efforts with thousands of photos [23, 2], or use a real-time tracking and mapping approach which is challenging to operate in a large space [5]. Either method makes it difficult for a single user to efficiently map a large, outdoor urban area.

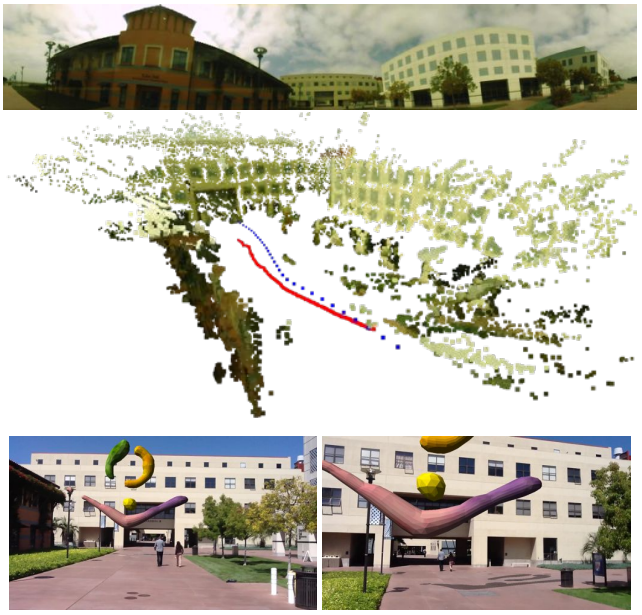


Figure 1: Overview of system operation. *Top:* Panorama from a thirty second video taken with a handheld omnidirectional camera while walking. *Middle:* Point cloud reconstruction produced in under fifteen minutes (densified for better visualization [8]). The blue points indicate panorama locations. *Bottom:* Two frames from an iPad video tracked in real-time with situated 3D graphics added. The camera track is plotted in red in the middle image.

This work aims to solve the bootstrapping problem for wide-area augmented reality with a fast and simple 3D mapping method using cheap, off-the-shelf hardware. In addition we describe and evaluate localization of a camera-equipped mobile device with respect to the map in real-time. Our system has three main components: offline 3D point cloud reconstruction from video taken with a consumer omnidirectional camera, processing of localization queries on a remote server, and real-time pose tracking on a mobile device.

1.1 Related Work

Previous works on outdoor mapping tend to neglect the issue of user effort in mapping a large space. For example, some works rely on manual capture of thousands of photographs from different viewpoints in the environment, as in the work of Irschara et al. [11] and Arth et al. [2]. An alternative is to download large image sets from a photo sharing website, as in the Photo Tourism system of Snavely et al. [23]; however, these types of reconstructions are typically lim-

*e-mail: jventura@cs.ucsb.edu
†e-mail: holl@cs.ucsb.edu

IEEE International Symposium on Mixed and Augmented Reality 2012
Science and Technology Proceedings
5 - 8 November 2012, Atlanta, Georgia
978-1-4673-4662-7/12/$31.00 ©2012 IEEE

ited to tourist sites where many people take and share photos. Similarly, driving through an area with a vehicle-mounted camera rig provides a large amount of coverage very quickly [1], but is still limited in range to city streets. Furthermore, this type of approach requires access to specialized equipment that most consumers do not have. In contrast to these approaches, we use a cheap omnidirectional camera held overhead to map an outdoor area in a few minutes. So far, the largest area we have mapped in one walking session is roughly 2000 m^2, but even larger areas can be handled by our system.

We compare our mapping procedure to previous works by comparing three measures: user effort, i.e. time spent creating a reconstruction, localization accuracy and tracking range. Overall our approach achieves a better balance of these three measures in comparison to previous localization systems such as that of Zhu et al. [26] and Arth et al. [2]. See Section 7 for a detailed description of our experiments.

Recent work by Pan et al. [21] is similar to ours in that their system allows a single user to capture panoramas, match them together, and produce a point cloud reconstruction of an outdoor environment. Their system uses a mobile phone which the user rotates to create each panorama; in addition, they adapt the modeling procedure to be performed on the phone itself. In our work, we use an omnidirectional camera which more quickly produces an accurate panorama; in addition, it is sufficient for us to relegate structure-from-motion processing to a server machine, which is not limited like the cell phone in terms of storage and computing power. Pan et al. also did not evaluate the potential of their maps for visual localization.

Some previous approaches use a polygonal model of the outdoor environment for localization and tracking, such as the system of Reitmayr and Drummond [22] and Karlekar et al. [12]. Production of such models is a time-consuming manual process. In contrast we use automated computer vision techniques to produce a 3D point cloud for localization.

In inspiring early work, Lee, You and Neumann used an omnidirectional camera to map a space and for real-time AR tracking [17]. In their system they use the omnidirectional camera for both mapping and tracking. They do not produce a 3D point cloud, but instead estimate the relative pose to two reference frames based on 2D-2D correspondences. This method has the downside of a degenerate case when all three cameras lie along a line. Overall, our system is more flexible because it does not require an omnidirectional camera on the tracked mobile device. Oe, Sato and Yakoza use a reconstruction from omnidirectional video to track a video feed, but do not consider a real-time approach to the system [20].

Many previous works using a server-client architecture for tracking are focused on 2D object recognition [24, 9] or are limited to homography-based tracking of flat objects [10]. Our system supports full 6DOF visual tracking of the entire surrounding enviroment.

Much success has been demonstrated recently for parallel tracking and mapping, where the system simultaneously estimates and tracks a 3D map of the immediate environment. Most notable is the Parallel Tracking and Mapping (PTAM) system by Klein and Murray [13] and its extension to multiple maps [5] and mobile phone implementation [15]. In this approach, as the user moves a handheld camera around the scene, the 3D map is built from scratch and used for tracking and annotation. While a powerful approach for small indoor scenes, this approach has limitations for larger outdoor scenes. The main issue is that the system requires careful movement of the camera to ensure enough baseline for 3D point triangulation, especially in the outdoor case. Furthermore, typical mobile devices are saddled with issues such as low camera field-of-view, rolling shutter, and limited computational resources, which make the live mapping problem especially difficult. For these reasons, we chose to use an offline mapping step which only requires the user to walk through the environment once while capturing images at all angles with the omnidirectional lens. This process is easy enough for an untrained user to perform, after which the 3D map is automatically produced.

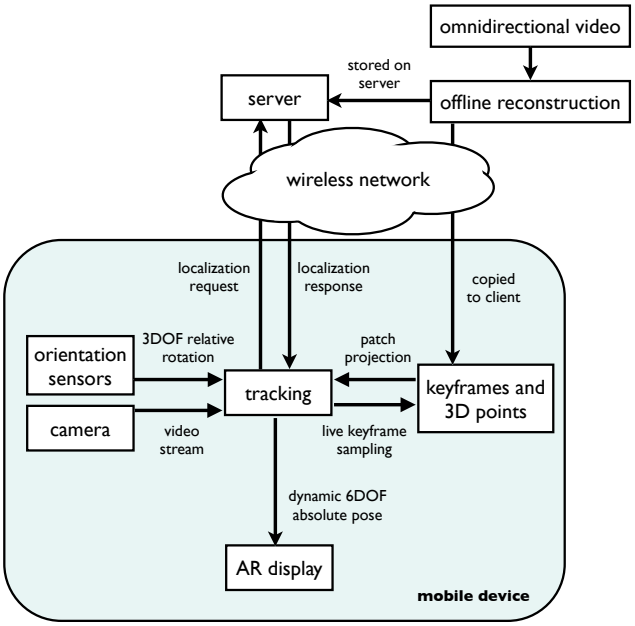


Figure 2: Overview of our system.

1.2 Contribution

The contribution of our work is a simple mapping system for outdoor environments which supports visual localization and tracking on mobile devices. Our system evaluation in several outdoor locations shows that little user effort is required to produce a map which achieves good tracking range and accuracy.

Our system combines visual and inertial sensors with cloud computing resources to enable ubiquitously available, accurately tracked augmented reality experiences. The result is a state-of-the-art mobile AR system which enables future areas of research and applications.

2 System Overview

Our approach has three main components:

- Construction of point-cloud models from omnidirectional video captured by a walking user
- Feature-based localization using a remote server to query the database
- Real-time tracking on a mobile device

See Figure 2 for a flowchart illustrating the system design and interactions between various components. Details about our mapping, localization and tracking methods are given in Sections 3 and 4. In Section 5 we consider how to improve tracker performance and reduce jitter. In Section 6 we analyze the problem of latency between client and server and methods to mitigate this latency for real-time operation. Finally, we present our system evaluation in Section 7 and conclusions in Section 8.

3 Outdoor Mapping

We create a point cloud reconstruction of the surrounding urban environment from a set of panoramas captured with an omnidirectional camera.

In our experiments we used the Sony Bloggie camera with a catadioptric lens attachment. A similar attachment is available commercially for the Apple iPhone as well. Such cheap, off-the-shelf camera solutions are light enough to be held by hand and do not require any extra equipment to record video.

3.1 Panorama Capture

To model building courtyards, we capture video with the omnidirectional camera held overhead in one hand while walking.

Because we perform offline, batch processing of the video, it is not as important as in an online SLAM system to ensure that the camera always translates as opposed to rotates. However if we can assume translation between consecutive frames, we can avoid costly pairwise matching of all frames in the sequence. For this reason we have found it advantageous to walk in a relatively straight path through the center of an environment. We also typically walk parallel to building walls, to maximize the range of perspective views of the facades.

3.2 Image Extraction

We extract panoramas from the video by sampling frames at regular intervals. The appropriate interval depends on the speed of motion and the distance to the buildings. For walking speed in building courtyards, we typically extract two panoramas per second.

Each omnidirectional image is warped into eight overlapping perspective views, with a wide horizontal field of view. The views are arranged at equal rotational increments about the vertical camera axis.

3.3 Offline Reconstruction Process

The perspective views from the panoramas are then processed in an incremental structure-from-motion pipeline which produces a 3D point cloud from image feature correspondences. This pipeline is adapted from that of Snavely et al. [23] with a few modifications for our scenario of panoramic video. We assume a linear camera path without loop closures, and thus only match panoramas to their neighbors in the sequence. Each panorama is matched to the four next panoramas in the sequence. We use the SIFT detector and descriptor for feature matching [18]. After triangulating points using an initial panorama pair, we incrementally add panoramas, triangulate more points, and perform bundle adjustment, until no more panoramas can be added. The relative rotation between perspective views in a single panorama is fixed, and only the rotation and translation between panoramas is estimated and refined. Each point's normal is set to the viewing direction of the first camera which observes the point.

3.4 Setting Orientation and Scale

It is useful for application purposes to register the reconstruction to a canonical frame of reference. In particular, we choose to translate the origin of the reconstruction to a point on the ground plane, rotate the reconstruction such that the Y axis is parallel to gravity, and scale the reconstruction to meter units. We perform this registration manually with a simple application which allows the user to identify two points on an edge of a building: one at the ground, and one at the roof. This manual step only has to be performed once per map. The user clicks to specify observations of the points in different images from the reconstruction. Once finished, we perform linear triangulation to estimate the two 3D points. The bottom corner point is translated to the origin, and the vector between the bottom and top corner is rotated to be the up vector (negative Y direction). This determines the ground plane which coincides with the origin and has normal parallel to the up vector. Also, the user can specify the distance between the two points, i.e. the height of the building in meters, to set the scale of the reconstruction.

4 Online Camera Localization

The online component of the system uses the keyframes and point cloud produced using the above modeling procedure for real-time localization of a camera-equipped mobile device. The localization component has two subsystems: initialization and tracking. Initialization is required when starting the system, and when re-starting after a tracking failure. Once initialized, continuous tracking updates the camera pose as new frames are grabbed from the image sensor.

4.1 Initialization

To initialize live camera tracking, we use a standard feature-based procedure which attempts to match features in the current camera image to 3D points in the model. We establish correspondences by matching features in the current camera image against triangulated features in the database. Each query feature is matched to its closest neighbor according to Euclidean distance between SIFT descriptors, and thus matched to the corresponding 3D point. Then we use the PROSAC procedure [6] for random sampling of correspondences to determine the best pose, using three 2D-3D correspondences to estimate an absolute pose [7].

For nearest neighbor queries we use brute force search, i.e. exhaustive comparison to all features in the database. In their experiments, Arth et al. [3, 2] have found exhaustive search to out-perform hierarchical k-means [19] in terms of finding geometrically correct matches. Brute force search can also be made reasonably fast by using parallel processing.

The feature-based initialization process can be performed either on the mobile device itself, or on a remote server or compute cloud. In the latter case, the camera image is compressed and sent over the network, and the server sends back the camera pose after processing. In our experiments we used a remote server which is able to process a localization query in about one to three seconds (see Section 7.4).

4.2 Tracking

The feature-based localization procedure is capable of determining the camera pose from a wide range of positions, but is clearly not fast enough for real-time operation. We complement the slow, but robust, initialization procedure with a fast patch tracking method which requires a pose prior.

The continuously operating tracker maintains in real-time the pose of the mobile phone or tablet with respect to the model. The tracker takes as input a pose prior and sensor readings, and outputs a pose posterior which estimates the current device position and orientation. The pose prior can be provided by the previous tracking iteration, or by feature-based localization when initializing or recovering from tracking failure.

At each frame, the tracker projects patches from the database of images into the camera frame according to the pose prior, searches for features in a window around their expected positions, and then updates the pose using gradient descent to minimize the reprojection error. Overall the tracking approach is similar to that of Klein and Murray [15]. The major difference is that we use a full perspective patch warp rather than an affine approximation. This is possible because we assume a moderate field-of-view camera with low radial distortion.

4.2.1 Patch Search

At each frame, a pose prior $\mathbf{P}$ is used to project each point $\mathbf{X_i} = (X_i, Y_i, Z_i, W_i)$ into the camera image, giving the projection point

$$\mathbf{x_i} = \text{proj}(\mathbf{KPX_i}). \quad (1)$$

The 3×3 matrix $\mathbf{K}$ represents the intrinsic camera calibration parameters. We use a single calibration parameter, the focal length f, and assume that the principal point lies at the image center. We also leave out radial distortion components and assume a camera with low distortion. The 3×4 matrix $\mathbf{P} = [\mathbf{R} \mid \mathbf{t}]$ describes the extrinsic parameters of the camera.

After culling points which are predicted to lie behind the camera or outside of the image, we search for the true location of the point's projection in the image. This correspondence is found by sampling an 8×8 pixel patch from a source image which observes the projected point $\mathbf{X}$ and searching for this patch in a region around the proposed location. The point and its normal define the plane $\pi = (n_1, n_2, n_3, D)$ from which we determine the patch warp. If the target and source projection matrices are $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P_i} = [\mathbf{R}_i \mid \mathbf{t}_i]$ respectively, the 3×3 perspective warp $\mathbf{W_i}$ from the camera image to the source image is [4]

$$\mathbf{W_i} = \mathbf{K}_{source}(\mathbf{R_i} + \mathbf{t_i}\mathbf{v}^\mathsf{T})\mathbf{K}^{-1} \text{ where } \mathbf{v} = -\frac{1}{D}(n_1, n_2, n_3)^\mathsf{T}. \quad (2)$$

The determinant of the warp $|\mathbf{W}_i|$ gives the amount of scaling between camera and source image. We compute this warp for all source images which observe the point, and choose the source image with scale closest to one. This ensures the best resolution when sampling the template patch. The system also chooses the best level of the source image pyramid according to resolution when sampling the template patch.

We search for the point's projected location by computing the normalized cross-correlation between the template patch and the target image at all locations on an 8×8 grid around the location given by the pose prior. The location with the best score is accepted if the score exceeds a threshold. In practice, we use a threshold of 0.7, which we experimentally found to adequately separate correct and incorrect matches. To increase robustness to fast movements, the search for correspondence is performed over an image pyramid of four levels.

4.2.2 Pose Update

After searching for correspondences, the camera pose estimate is updated to fit the measurements. All correspondences found during patch search are used to update the camera pose, even if they were not successfully refined to the lowest pyramid level. For each observed point, we project its search location down to the zero pyramid level, giving a measured location $\tilde{\mathbf{x}}_i$ for point $\mathbf{X}_i$. We use ten iterations of gradient descent to minimize the re-projection error

$$e = \sum_i m(|\tilde{\mathbf{x}}_i - \mathbf{x}_i|^2). \quad (3)$$

We weight the measurements using a Tukey M-estimator, recomputing the weights during the first five iterations [13].

4.2.3 Success Metric

After the pose update, we count the number N_{found} of points found to be inliers by the M-estimator. This is an indicator of the success of the tracker, i.e. whether the pose posterior matches the true pose of the camera. We use the threshold ratio $\tau = N_{found}/N_{attempted}$ to determine whether to accept the pose posterior, where $N_{attempted}$ is the number of points searched for. In practice we have found that $\tau < 30\%$ is an acceptable threshold to identify when a tracking result should be rejected. We also require that at least 200 points have been successfully tracked ($N_{found} \geq 200$).

5 Performance Considerations

We describe here some considerations with respect to increasing the speed of the tracker and reducing jitter in the estimated pose.

5.1 Tracker Point Selection

To ensure an acceptable frame rate, we place a limit N_{max} on the number of points $N_{attempted}$ which the tracker can attempt to find in a single frame. In practice we use $N_{max} = 1024$.

The system first performs view frustum culling on all points, and then selects $N_{attempted} \leq N_{max}$ points to search for, using the maximum if more than N_{max} points are visible.

The point selection is performed by choosing some ordering of visible points, and then selecting the first N_{max} points to be tracked. The question is, what is a suitable ordering for best tracking performance?

Previously Klein and Murray [13] proposed using a random ordering, i.e. randomly shuffling all visible points at each frame. However, we found that this can lead to pose jitter when the camera is not moving. The reason is that using different subsets of the points for tracking may result in slightly different poses found by the gradient descent minimization, because of slight errors in patch search.

Our solution to this problem is to randomly order all points once, at system startup. This provides a fixed, but random, ordering for the points at each frame. The result is that for a static or slowly moving camera, the tracker will reach a steady state where the same subset of points is used for tracking and pose update at each frame. Overall we found that this sampling procedure reduces pose jitter in comparison to producing a new random ordering of points at each frame.

5.2 Live Keyframe Sampling

A second source of pose inaccuracy is poor feature correspondence. Errors in the patch search can prevent the pose update from correctly converging.

We have found that direct alignment of the mobile device camera image to the panorama keyframes can cause poor feature correspondence which leads to inaccurate or jittery pose estimates. This is most likely because of the difference in imaging characteristics of the two cameras, such as focal length, and sharpness. In contrast, the feature correspondence between two nearby images from the same camera is less noisy. Previously, Vacchetti, Lepetit and Fua demonstrated that matching to recent frames as well as training images improves jitter in camera tracking with few, wide-baseline keyframes [25]. Our insight is that keyframe sampling also reduces jitter when we have sufficiently dense training images, but those training images were captured with a different imaging system and different environmental conditions.

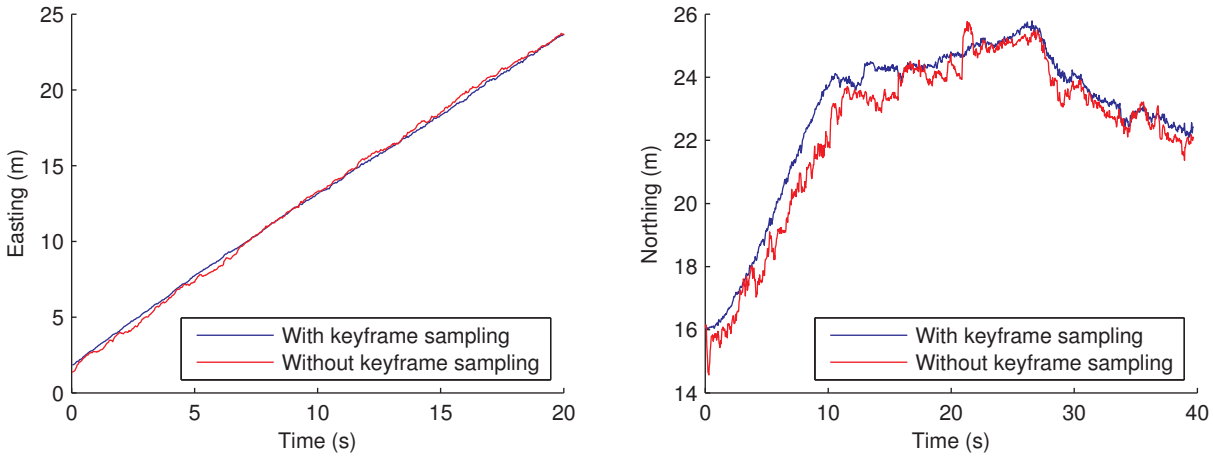
This insight led us to incorporate live keyframe sampling into the tracking system. During tracker operation, we collect keyframes from the current video stream and add them to the set of images used for patch projection. We preferentially project patches from the new keyframes, as these lead to more stable pose estimation.

The decision of when to sample a new keyframe is based on the number N_{old} of points which are projected from a panorama keyframe in the current camera image, and the number N_{new} of points projected from a new keyframe. When the ratio N_{new}/N_{old} drops below 50%, or when the distance to the nearest new keyframe rises above 2 meters, we sample a new keyframe, associating the inlier measurements to their 3D points to be used for future patch projection.

In Section 7.3 we evaluate the effect of live keyframe sampling on pose accuracy and jitter.

6 Mitigating Latency

Due to network communication time and server computation time, feature-based initialization introduces latency between an image query and a 6DOF pose response. During this time, the camera might be moved from its query position, introducing error in the



(a) Walking east along a straight line. We fit a least-squares line to each path; RMS with sampling: 0.1361 m. RMS without sampling: 0.2100 m.

(b) Freely walking.

Figure 3: Evaluation of pose accuracy with and without live keyframe sampling. Projecting and tracking patches from the live video stream, instead of the panorama keyframes, increases pose accuracy and reduces pose jitter.

localization pose estimate. Thus, the system needs some ability to handle an out-dated pose estimate from the localization system.

6.1 Latency Analysis

The amount of error which the system can tolerate is determined by the region of convergence of the tracker. The continuous pose tracker uses a patch search method to find a point given a pose prior. This search occurs over a fixed region around the estimated point projection location, and is run over an image pyramid to expand the search region. This establishes a maximum pixel error in the projected point location which will still lead to tracker convergence.

We use a simplified analysis here by considering movement in one dimension, to produce an estimate of the tracker convergence region.

Assuming rotation around the Y-axis (vertical axis), a rotational error of θ_{err} degrees will cause a pixel offset of x_{err} pixels:

$$x_{err} = f \cdot \tan(\theta_{err}) \tag{4}$$

where f is the focal length parameter of the camera's intrinsic calibration matrix. The maximum projection error can be used to find the maximum rotational pose error θ_{max}.

Our system uses an effective search radius of $4 \times 2^3 = 32$ pixels, and the iPad 2 camera has a focal length of $f = 1179.90$. Thus, the maximum rotational pose error is $\theta_{max} = 1.55$ degrees. We estimate that this limit could be a problem if localization latency is one second or more.

For the translation case, the maximum translation t_X depends on the distance Z to the observed object:

$$x_{err} = f \cdot \frac{t_X}{Z}. \tag{5}$$

Using the numbers given above for our system, we find a maximum translation of $t_X/Z = .03$. Given a building that is 12 meters away, the maximum translation would be about 1/3 meter. This as well would be a limitation for localization given a fast-walking user.

6.2 Sensor Integration

To overcome the problem of rotational movement during the latency period, we integrate gyroscope rotation measurements to estimate the orientation of the device with respect to the query frame. The integrated rotation is applied to the localization response before attempting to initialize the tracker.

A similar approach could be applied to estimate translation based on accelerometer readings. However, the accelerometer found in typical consumer devices such as the iPad 2 are too noisy for estimating translation over a period of several seconds. Fortunately, translational error during the latency period is not an issue in larger environments such as typical urban scenes.

6.3 Fast Recovery

A second solution to the problem of client-server latency is to avoid remote localization queries altogether, by using a faster (but less robust) localization method on the mobile device itself. We implemented a fast recovery method inspired by that of Klein and Murray [14]. The concept of the method is to find a previously-seen image which is close in appearance to the current camera image, and try re-initializing to the cached pose before querying the server and waiting for a localization response.

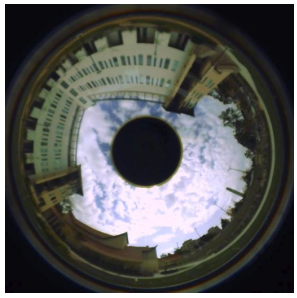
The system maintains on the mobile device a cache of images with known pose, gathered during a tracking session. A tracked image is added to the cache when tracking is successful and the closest keyframe in the cache is more than 1 meter away or 45 degrees away.

When tracking fails and the system enters a lost state, the camera image is first compared to every keyframe in the cache. We compare images at the fifth pyramid level using normalized cross correlation. The pose of the image with the highest correlation score is used as initialization for the tracker. If tracking succeeds, we can resume tracker operation without requesting a slower, feature-based localization on the server.

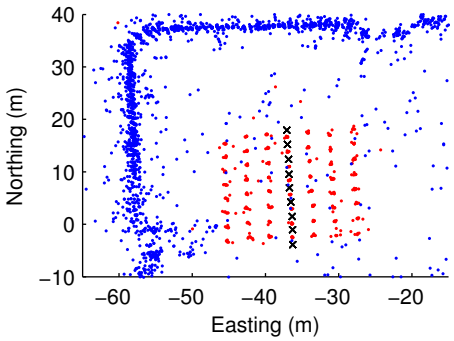
7 System Evaluation

We evaluated several important aspects of our modeling and localization system. One goal of our investigation is to determine the relationship between modeling effort and localization performance. A second goal is to evaluate modeling, localization and live augmentation in realistic use cases.

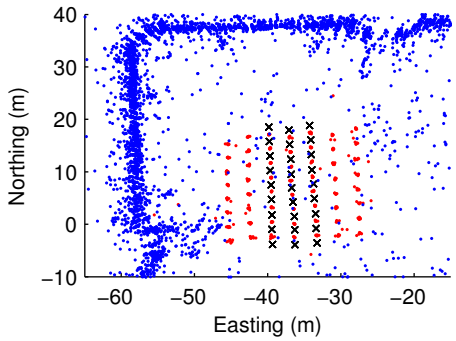
(a) Aerial photograph of the testing area.



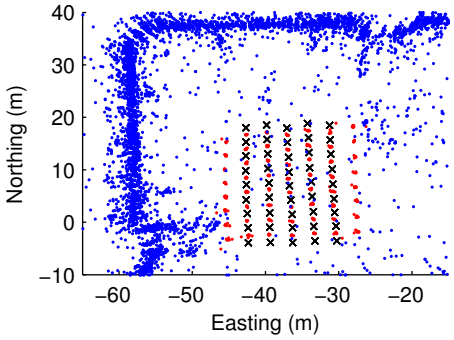
(b) One of the training panoramas.



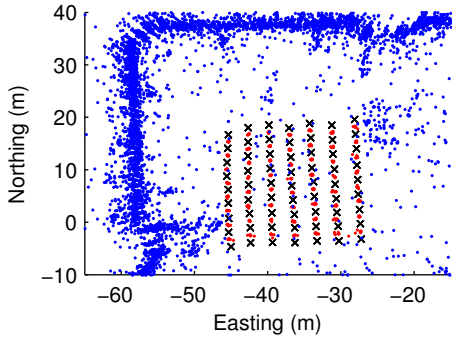
(c) Results with training column 4.



(d) Results with training columns 3-5.



(e) Results with training columns 2-6.



(f) Results with training columns 1-7.

Figure 4: Overview of our localization test. The blue points are triangulated 3D points, the black X's are training panoramas, and the red dots are localized test images. The panoramas are organized into columns numbered one to seven from left to right.

7.1 User Modeling Effort

The intention of our system design is to allow a single user to quickly and easily capture a 3D model of an outdoor space, to be later used for wide-area tracking on a mobile device. To validate the suitability of our design, we devised an experiment to relate localization accuracy to modeling effort, i.e. time spent capturing panoramic video.

7.1.1 Grid Dataset

We created a test dataset of panoramic videos in a building courtyard with two large perpendicular walls. We captured panoramas by walking with the omnidirectional camera along straight lines parallel to one wall. The result is a grid of panoramas in the middle of the building courtyard, illustrated in Figure 4.

To produce a ground truth location for each panorama, we ran all panoramas through our structure from motion pipeline, which estimates positions of the cameras and triangulates 3D points. The estimated camera position of each panorama is used as the ground truth location in our tests. The distance between the lines was physically measured on the ground to be 3.11 meters. This measurement determined the global scale of the reconstruction.

From each panorama we produced eight perspective images at equal increments about the vertical axis. Altogether the dataset has seven columns and seventeen rows each, giving 119 panoramas, or 952 images total.

We split the panoramas into a training set and a testing set by assigning the odd-numbered rows to the training set and the even-numbered rows to the testing set. This gave nine training panoramas and eight testing panoramas per column.

7.1.2 Localization Accuracy and Range

We tested the effect on localization of the testing panoramas when using subsets of the training panoramas. Our hypothesis was that localization queries which are farther from the training video capture location would be less accurate.

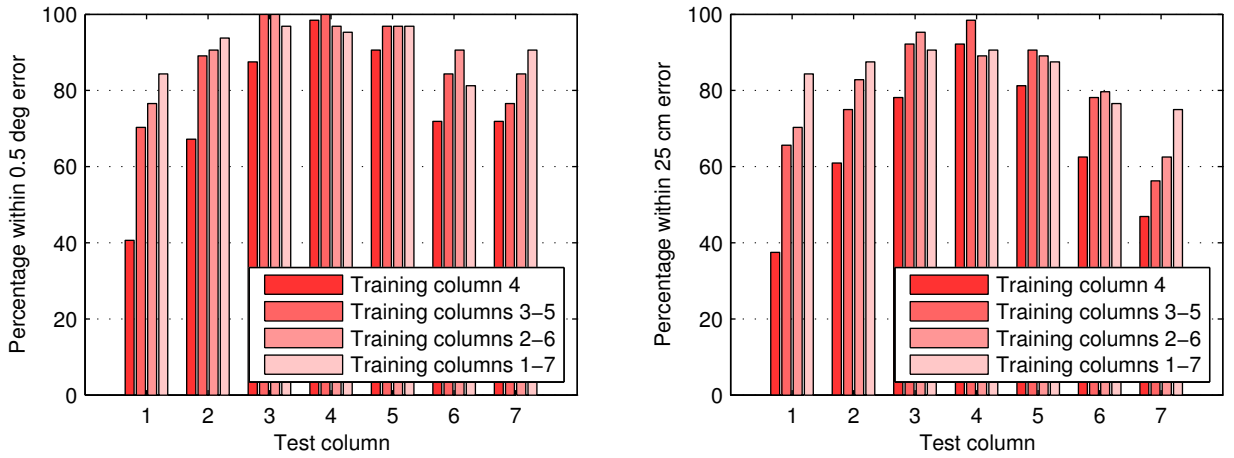
We used a subset of one video (column four), three videos (columns three to five), five videos (columns two to six) and seven videos (columns one to seven). The video recording time is our measure of user effort. Each video was 30 seconds long, so the conditions correspond to 30, 90, 150, and 210 seconds of video.

For each subset, we then ran the localization algorithm on every image in the training set, but restricted the features and 3D points in the database to those visible in the chosen training subset. In conditions with fewer training panoramas, we are testing the ability to localize when the query camera is far away from the video capture location.

The training subsets and localization test results are plotted in Figure 4. The localization results are more noisy in the conditions with fewer training panoramas. Figure 5 plots the percentage of images in each testing column and each condition which are accurately localized to acceptable position and orientation error thresholds.

We used thresholds of 0.5 degrees orientation error and 25 cm positional error in our evaluation. In total, the percentage of images localized within the error thresholds was 62.5% with one video, 77.46% with three videos, 79.24% with five videos and 81.92% with seven videos.

These numbers give an indication of tracking accuracy and range given user effort. If we map an area with one line of panoramas, we can expect to localize well within three meters of the line, and to have slightly poorer but still acceptable performance up to nine me-



(a) Percentage of images localized within 0.5 deg angular error

(b) Percentage of images localized within 25 cm translational error

Figure 5: Results of localization tests using the camera grid illustrated in Figure 4. We tested localization of each image by querying different subsets of the training images. Overall, the localization accuracy improves as we add more training images.

ters away. If we spend more time mapping the environment by also capturing panoramas three meters away from the original line in either direction, we expect the localization reliability to be improved in the same range. Adding more panoramas beyond this does not have as significant of an effect according to our experiment. We expect these numbers to be proportional to the distance to the building walls; in our experiment the nearest walls were about 40 to 60 meters away.

Our system compares favorably to previous systems in terms of the balance between mapping effort, localization accuracy and range. Zhu et al. report accuracy under 25 cm error by sampling images at two meter increments [26]. They use a multi-stereo helmet with four cameras for mapping and localization. In their tests, Arth et al. report accuracy under 25 cm positional error and 1-2 degrees rotational error. They use a large-scale reconstruction created from thousands of photographs, partitioned into approximately 800 m^2 blocks.

Location	Video	# Panos.	# Points	Reconst.
Media Studies	90 s	27	5890	488 s
Phys. Sciences	33 s	17	3471	475 s
Kirby Crossing	33 s	31	4481	832 s
Girvetz Hall	29 s	29	4392	770 s

Table 1: Details about our reconstructions of various urban scenes.

Location	Approx. Area	Test Videos	Tracked
Media Studies	33 m × 33 m	183 s	95%
Phys. Sciences	50 m × 40 m	175 s	83%
Kirby Crossing	60 m × 8 m	106 s	52%
Girvetz Hall	25 m × 15 m	52 s	45%

Table 2: Percentage of frames tracked, using three seconds of localization latency.

7.2 Realistic Capture Conditions

In addition to the grid dataset described above, we created and processed panoramic videos in several other campus locations. For these videos we simply walked through the environment while holding the omnidirectional camera overhead. Details about these reconstructions are given in Tables 1 and 2. Figure 6 gives an overview of one such on-campus environment, the Physical Sciences courtyard.

We also captured several test videos using the iPad 2 camera in these locations. For the test videos, we walked around the environment while aiming the camera generally at the nearby buildings. We then tested the ability of our system to localize and track these videos. The approximate area where localization was successful is given in Table 2.

For this test we pre-processed the videos by attempting to localize every 90th frame of each video, to represent a latency of three seconds. Then we processed each video with our tracking software, which only used a pre-processed localization result after 90 frames had passed. This simulated the effect of three seconds of latency.

Table 2 gives the percentage of frames succesfully tracked for each location. For smaller areas where the camera is closer to building walls, the latency has a more negative effect on tracking, because translating the camera causes more displacement in the image. Smaller areas are also challenging because the iPad 2 camera has a narrow field of view, and thus in some cases does not see enough features to make localization possible.

Depending on the environment, we achieve between 45% and 95% frames tracked. However, this measure does not take into account the accuracy of the estimated camera pose, since for these tests we do not have ground truth position data.

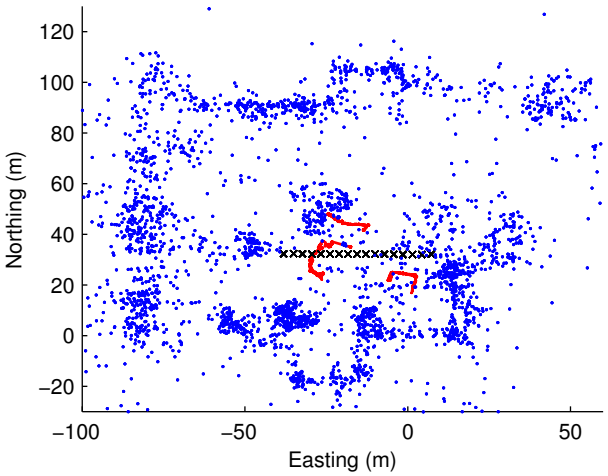
7.3 Tracking Accuracy and Jitter

The live keyframe sampling method introduced in Section 5.2 is intended to reduce jitter in the tracker pose. We assessed this method by tracking the same videos both with and without live keyframe sampling. Figure 3 shows plots of the estimated path of the camera for both videos.

In Figure 3a we plot the estimated path of the camera when walking east along a straight line. For each path we performed a least-squares linear fit and calculated the RMS error. The estimated path with keyframe sampling results in less jitter: with keyframe sampling, the RMS is 0.1361 m; without keyframe sampling, the RMS is 0.2100 m.

Figure 3b plots the estimated path of a video where we freely walked in an environment. In this video as well, the tracker pro-

(a) Aerial photograph of tracking area



(b) Top-down view of reconstruction and camera tracks

Figure 6: Overview of a campus environment mapped using our system. Black crosses mark panorama locations used to map the environment, with 3D points shown as blue dots. Red lines plot camera paths produced from mobile tracking sessions. The total tracking area is approximately 2000 m^2.

Localization Timing	
Image transfer	35 ms
Feature extraction	896 ms
Nearest neighbors	206 ms
Robust pose estimation	466 ms
Total	1603 ms

Table 3: Average timings for localization on a remote server. Timings are rounded averages over ten trials for a model with 21 panoramas, 3691 points and 6823 features. We measured image transfer time using a 3G cellular data connection outdoors.

duces a smoother path when keyframe sampling is enabled.

7.4 Timing

An important requirement for our system is to support online operation at a real-time rate. The two most time-consuming components are localization (performed on a remote server) and tracking (performed on the device).

7.4.1 Localization

Localization queries are processed on a remote server while the mobile tracker continues running. This means that the server does not have to respond within 33 ms, since the processing happens in the background. However, the processing time should be as low as possible to provide a smooth tracking experience.

The average timings for our implementation are given in Table 3. We used a Mac Pro with a 2.26 GHz Quad-Core Intel Xeon and 8 GB RAM. The model we tested has 21 panoramas, 3691 points and 6823 features.

For a model size, the most computation time is spent on SIFT feature extraction (900 ms) and PROSAC pose estimation (500 ms). As the number of features in the database grows, the brute force nearest neighbors computation increases and can significantly impact localization time.

The time to transfer a JPEG-compressed image from the device to the server is not a severe bottleneck, even with a 3G cellular data connection. Transfer time typically takes 30-40 ms using either a wireless or 3G connection.

Overall, with our test we calculated an average localization latency of about one and a half seconds. In practice we have experienced localization times of two to three seconds for a larger model.

7.4.2 Tracking

Feature-based tracking consists of three steps which constitute the majority of computation time per frame: point culling (.005 ms per point); patch warp (.02 ms per point); and patch search (.033 ms per point). The total tracking time per frame depends on the total number of points in the model N_{total}, the number of points tracked N_{track}, and the number of pyramid levels L. This gives an approximate tracking time per frame:

$$t_{track} = N_{total} \cdot t_{cull} + N_{track} \cdot L \cdot (t_{warp} + t_{search}) \quad (6)$$

With multi-threading on the dual core iPad 2, the processing time is approximately reduced by half. For a model with 3691 points, 1024 tracked points and four pyramid levels, this gives a maximum tracking time of approximately 117 ms per frame. However, typically the number of points tracked decreases at each successive pyramid search level, so the actual tracking time in practice is lower. We typically record 15-20 frames per second tracking on the mobile device.

7.5 Live Augmentation

We implemented a prototype augmented reality system on the iPad 2 using our tracking methods. Once tracking is initiated, the user can touch the screen to place a 3D object. The object position is set by intersecting the ray from the finger to the ground plane. The system then renders the 3D object with a shadow on the ground.

Sample tracked camera frames with augmented 3D graphics are presented in Figures 1, 7 and 8. Figure 7 depicts a landscape design application where users can place virtual trees on the grass to evaluate their appearance. Figure 8 depicts an entertainment application where a virtual spaceship hovers overhead.

Note that the tracking system allows for both rotation and translation, meaning that the user can freely walk around the environment while maintaining 3D pose tracking. The only requirement

Figure 7: Prototype landscape design application. The user moves towards the grass while viewing virtual trees.

Figure 8: Prototype entertainment application showing a virtual spaceship in a building courtyard.

is that the buildings be kept at least partially visible in the camera frame.

We also tested mapping and localization along a city street (Branch Street in Arroyo Grande, CA). To create the reconstruction we mounted the omnidirectional camera above a car and drove down the city street while recording video. Localization was successful with iPad 2 videos recorded while walking down the sidewalks lining the street. The reconstruction and a sample tracked image are presented in Figure 10.

7.6 Discussion

In our evaluations we have demonstrated mapping, localization and tracking in a range of outdoor environments. In a roughly 1000 m^2 area, we achieved positional accuracy below 25 cm error, and rotational accuracy under 0.5 degrees for above 80% of test images. It should be noted that in this test we used the panorama keyframes as input to the localization process, whereas in our live demonstrations we use the iPad 2 camera. The iPad 2 camera, like most mobile devices, has a narrower field of view ($\sim$ 45 degrees) in comparison the panorama keyframes ($\sim$ 125 degrees). Arth et al. demonstrated that a larger field of view leads to more reliable localization results [2]. However, due to perspective projection there are not many features extracted at the extremities of the panorama keyframes, so in practice we only experience slightly worse performance using iPad 2 input imagery.

In our experiments we have encountered many scenarios where localization fails in outdoor scenes. For example, in the Physical Sciences courtyard (Figure 6), the building to the south lack much texture and are challenging for the system to recognize. Some further examples are depicted in Figure 9. The most common problems are: repetitive structures such as doors and windows, which confuse the feature matching; and occluding objects such as trees and bushes. Such natural objects also generate many confusing features because of their high textured-ness. These issues are compounded by the narrow field-of-view camera image, which restricts visibility of the scene when close to buildings. Some recent work has focused on alleviating these issues. For example, Arth et al. propose increasing the field of view of the query image by panorama stitching [2], and Knopp et al. propose detecting and removing confusing features [16].

Another issue we have encountered is the effect of false positives reported by the patch-based tracker. The normalized cross-correlation score can erroneously report a high matching score when viewing a low-contrast scene. This sometimes causes the system to initialize tracking with an incorrect pose. This is especially a problem when using live keyframe sampling, which assumes that the estimated pose is correct; incorporating an incorrectly tracked keyframe leads to corruption of the image database. We believe this problem could be mitigated by adding more features into tracking, such as edges [14], and by cross-checking the estimated pose with other sensor modalities such as a digital compass and GPS.

8 Conclusions and Future Work

Although general, wide-area tracking and information overlay has been part of the augmented reality vision since its inception, few works to date have tackled this difficult problem. Most outdoor tracking systems are limited to a fixed position, only provide low accuracy 6DOF tracking from GPS, or are restricted to bulky or specialized hardware such as wearable computers and well-instrumented helmets. Our system offers an alternative which uses commonly available hardware such as a smartphone or tablet and provides accurate 6DOF tracking in real-time. The result is a flexible mobile tracking system for augmented reality which allows users to freely move and orient themselves in an outdoor space.

We believe our approach is useful as a way to bootstrap wide-area tracking in an unprepared outdoor environment. We envision a future system which builds upon the map by collecting more imagery as users interact with the AR application. Such data collection, after integrated into the map, would improve the AR experience for later users. For example, we showed in our evaluation that live keyframe sampling improves tracking stability in a single session; ideally the system would store these keyframes and integrate them into the map to later be used again. Or, images collected at different times of day or other year could be collected and integrated to improve localization robustness across illumination and scene changes. Our mapping system provides a crucial first step for growing an extensive visual map which enables widely-available augmented reality applications.

Acknowledgements

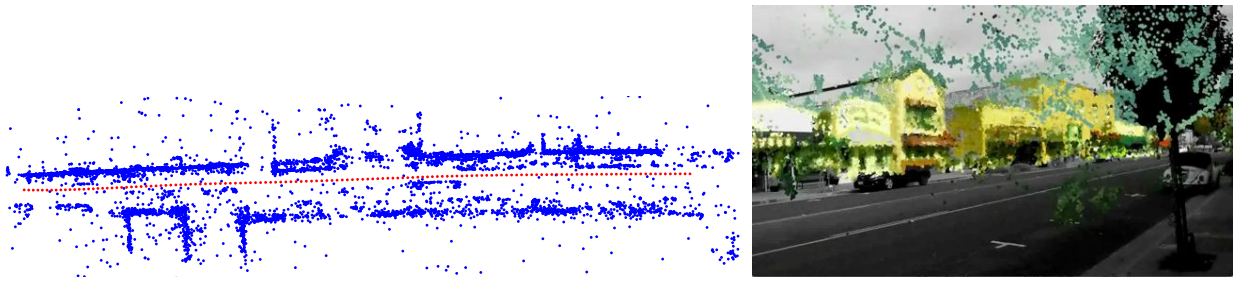
This work was partially supported by NSF CAREER grant IIS-0747520 and ONR grant N00014-09-1-1113.

References

[1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google Street View: Capturing the World at Street Level. *Computer*, 43(6):32–38, 2010.

[2] C. Arth, M. Klopschitz, G. Reitmayr, and D. Schmalstieg. Real-Time Self-Localization from Panoramic Images on Mobile Devices. In *International Symposium on Mixed and Augmented Reality*, pages 37–46. IEEE, 2011.

Figure 9: Examples of images which are difficult to localize. Problems such as repetitive structures and occluding foliage are compounded by the low field-of-view camera image.



(a) Overhead view of city street reconstruction

(b) Tracked image showing point cloud overlay

Figure 10: Reconstruction and tracking along a city street. We mounted the omnidirectional camera above a car and drove down the city street to collect images. The right image depicts overlay of the dense point cloud (in color) on the tracked mobile device camera image (in grayscale).

[3] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *International Symposium on Mixed and Augmented Reality*, pages 73–82. IEEE, 2009.

[4] C. Baillard and A. Zisserman. A plane-sweep strategy for the 3D reconstruction of buildings from multiple images. *International Archives of Photogrammetry and Remote Sensing*, 33(B2; PART 2):56–62, 2000.

[5] R. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *International Symposium on Wearable Computers*, pages 15–22. IEEE, 2008.

[6] O. Chum and J. Matas. Matching with PROSAC-progressive sample consensus. In *Computer Vision and Pattern Recognition*, pages 220–226 vol. 1. IEEE, 2005.

[7] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[8] Y. Furukawa, B. Curless, and S. Seitz. Towards internet-scale multi-view stereo. *Computer Vision and Pattern Recognition*, 2010.

[9] S. Gammeter, A. Gassmann, L. Bossard, T. Quack, and L. Van Gool. Server-side object recognition and client-side object tracking for mobile augmented reality. *Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8, 2010.

[10] J. Ha, K. Cho, F. A. Rojas, and H. S. Yang. Real-time scalable recognition and tracking based on the server-client model for mobile Augmented Reality. *International Symposium on VR Innovation (ISVRI)*, pages 267–272, 2011.

[11] A. Irschara, C. Zach, J. M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Computer Vision and Pattern Recognition*, pages 2599–2606. IEEE, 2009.

[12] J. Karlekar, S. Zhou, W. Lu, Z. Loh, Y. Nakayama, and D. Hii. Positioning, tracking and mapping for outdoor augmentation. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 175–184. IEEE, 2010.

[13] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality*, pages 225–234. IEEE, 2007.

[14] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *Computer Vision–ECCV 2008*, pages 802–815. Springer, 2008.

[15] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *International Symposium on Mixed and Augmented Reality*, 2009.

[16] J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. *Computer Vision–ECCV 2010*, pages 748–761, 2010.

[17] J. Lee, S. You, and U. Neumann. Tracking with omni-directional vision for outdoor AR systems. In *International Symposium on Mixed and Augmented Reality*, pages 47–56. IEEE, 2002.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[19] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition*, pages 2161–2168. IEEE, 2006.

[20] M. Oe, T. Sato, and N. Yokoya. Estimating camera position and posture by using feature landmark database. *Image Analysis*, pages 257–260, 2005.

[21] Q. Pan, C. Arth, G. Reitmayr, E. Rosten, and T. Drummond. Rapid Scene Reconstruction on Mobile Phones from Panoramic Images. In *International Symposium on Mixed and Augmented Reality*, pages 55–64, Jan. 2011.

[22] G. Reitmayr and T. Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *International Symposium on Mixed and Augmented Reality*, pages 109–118. IEEE, 2006.

[23] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics (TOG)*, 25(3):835–846, 2006.

[24] 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. *International Conference on Multimedia Information Retrieval*, pages 427–434, 2008.

[25] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391, 2004.

[26] Z. Zhu, T. Oskiper, and S. Samarasekera. Real-time global localization with a pre-built visual landmark database. In *Computer Vision and Pattern Recognition*, 2008.