8 Urban Visual Modeling and Tracking

Jonathan Ventura and Tobias Höllerer

CONTENTS

8.1 INTRODUCTION

This chapter explains how to digitally capture, model, and track large outdoor spaces so that they can be used as environments for mobile-augmented reality (AR) applications. The three-dimensional (3D) visual model of the environment is used as a database for image-based pose tracking with a handheld camera-equipped tablet. Experimental analysis demonstrates that real-time localization with high accuracy can be achieved from models created using a small panoramic camera.

Device positioning is a common prerequisite for many AR applications. Indoors, visual detection of flat, printed markers has proven to be a very successful method for accurate device positioning, at least for a small workspace. In larger spaces, external tracking systems allow for precise positioning by use of statically mounted cameras that observe objects moving in the space. Outdoors, however, we cannot require that the environment be covered in printed markers or surrounded by mounted cameras. The global positioning system (GPS) provides ubiquitous device tracking from satellites, but does not guarantee enough accuracy on consumer-level devices for AR applications. This chapter presents an alternative approach that treats the built environment like an existing visual marker. By detecting and tracking landmark features on the building facades, the system uses the surrounding buildings for accurate device positioning in the same way that printed markers are used indoors, except at a larger scale.

Visual modeling and tracking technology is based on the relationship between points in the scene and cameras that observe them. Having images of the same point from multiple known camera positions allows us to determine the 3D location of the point, as depicted in Figure 8.1. Conversely, observing multiple known points in a single image allows us to determine the position of the camera, as depicted in Figure 8.2. The first case is useful for building a 3D model of an environment. The second case is useful for determining the location of a camera with respect to that model. Researchers in the fields of photogrammetry and multiple-view geometry have studied the equations and principles governing these relationships extensively. This chapter describes a system that applies these principles to model a large outdoor space and track the position of a camera-equipped mobile device moving in that space.

The process for preparing an outdoor environment for tracking usage in AR applications involves three basic steps. First, the area is captured in many photographs that cover the environment from all possible viewpoints. Second, from this collection of photographs, feature points are extracted and matched between images, and their 3D positions are precisely determined in an iterative process. Third, the 3D points are aligned with a map of building outlines to provide the reconstruction with a scale in meters and a global position and orientation.

The resulting 3D reconstruction is stored on a server and transferred to the client device. After computing a tracker initialization on the server, the software on the mobile client device tracks 3D feature points in the live camera view to continuously determine the position and orientation of the device.

The following sections provide detailed descriptions and evaluations of these system components. Sections 8.2 through 8.4 cover the outdoor modeling process. Sections 8.5 through 8.7 describe the outdoor tracking system. Section 8.8 provides

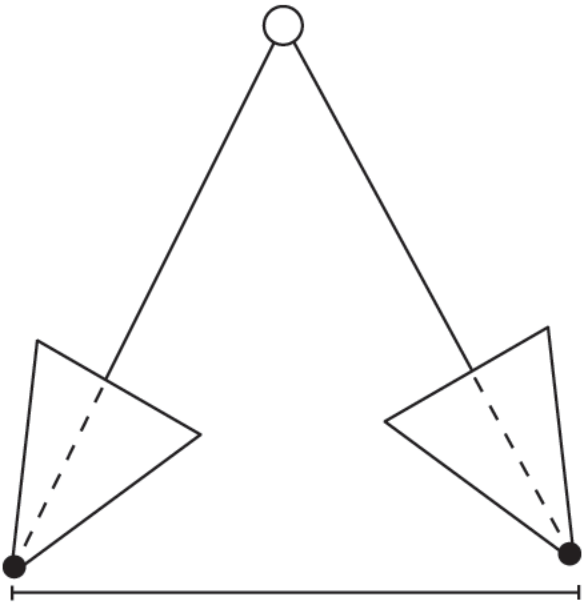

FIGURE 8.1 Triangulation of a 3D point from observations in two cameras. The distance between the cameras is called the *baseline*; the estimate is more accurate with a larger baseline.

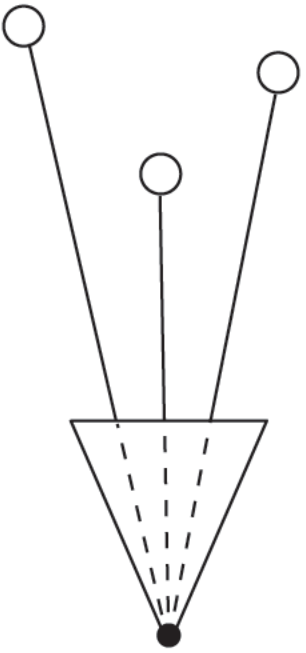

FIGURE 8.2 Localization of a camera from three 3D point observations. The estimate is generally more accurate when the points are nearer to the camera.

quantitative evaluations of the system, and Section 8.9 provides a discussion of the overall system design and performance. Finally, Section 8.10 gives an annotated reference list for interested readers who would like to explore related work.

8.2 OUTDOOR PANORAMIC CAPTURE

An easy way to capture many different viewpoints of a large environment is to use a panoramic or omnidirectional camera. Such a camera captures all viewing angles from one position in a single capture. Examples of such cameras include the professional-grade Point Grey Ladybug, which has six cameras, and the consumer-grade Ricoh Theta, which has just two cameras, placed back to back, with very wide-angle lenses.

A consumer-grade panorama camera is light enough to be held overhead in one's hand. Alternatively, a tripod or monopod attached to a backpack serves as an easy mounting system, if the camera can be remotely triggered.

8.2.1 Guidelines for Capture

By simply walking through an environment and capturing panoramas at regular intervals, the environment to be augmented is easily captured in images. However, some care should be taken during the capture process, in order to ensure the success of the later reconstruction and augmentation steps.

The main issue to consider is how many pictures to take, and where to take them. To answer this, the characteristics of vision-based 3D reconstruction should be taken into account. Triangulation of 3D points depends on multiple observations of a point taken from images in different locations, as shown in Figure 8.1. The distance between two cameras is called the *baseline*. A larger baseline gives a more accurate point triangulation. However, if the images are too far apart, then the appearance of the object will change too much, which means that the images cannot be automatically matched together. The scale-invariant feature transform (SIFT) descriptor (Lowe, 2004), which we use for matching, is reported to match well with up to 45° of out-of-plane rotation. However, the matching works better with smaller angles.

A simple rule of thumb is that the optimal distance ratio is about 4/10, meaning that the pictures should be about 4 m apart if the buildings are 10 m away. This corresponds to having an angle of about 10° between the rays observing a point.

When recording panoramic video, images are extracted at a fixed rate in order to have regularly spaced panoramas from the video to use in the reconstruction pipeline. The appropriate interval to achieve the desired 4/10 ratio depends on the speed of motion and the distance to the buildings. For walking speed in building courtyards, an appropriate sampling rate is about two panoramas per second.

A second consideration is the expected distance between the offline-captured panorama images and the user's location during online use of the AR application. Again, the limiting factor is the ability of the image points to be matched. This depends on both the angle of view and the change in scale. Experiments with the iPad 2 and the Point Grey Ladybug camera have shown that reasonable localization performance can be expected in a range within a quarter of the distance to the buildings from the offline panorama capture point (Ventura and Hollerer, 2012b).

8.3 AUTOMATIC 3D MODELING

After canvassing the area to be modeled and collecting imagery from many viewpoints, the image collection is processed in an automatic 3D modeling pipeline. This pipeline takes the image sequences as input and outputs the estimated camera positions and 3D triangulated points. The collection of estimated points is called a point cloud. This section gives some details about how the pipeline works.

8.3.1 Image Extraction

There are several common panoramic image representations that could be used to store the image sequences. Mappings such as spherical and cylindrical projection offer a continuous representation of all camera rays in one image. However, they nonlinearly distort the perspective view, which would impact performance when matching to images from a normal perspective camera, as found on a typical mobile device.

Instead of using spherical or cylindrical projection, perspective views are extracted from each panorama, such that the collection of extracted views covers the entire visual field. A typical cube map used as an environment map in rendering uses six images arranged orthogonally, with 90° horizontal and vertical fields of view in each image. This representation offers perspective views without distortion. However, in practice, the low field of view in each image hinders the matching and reconstruction process.

To address this issue, perspective images with wider than 90° horizontal field of view are used. The top and bottom of the cube are omitted, since they generally have no usable texture. The faces provide overlapping views, which increases the likelihood of matching across perspective distortion. Eight perspective views per panorama are used to increase image matching performance by ensuring that all directions are covered in a view without severe perspective distortion. The views are arranged at equal rotational increments about the vertical camera axis. Figure 8.3 shows an image from the panorama camera and its extended cube map representation.

(a)

FIGURE 8.3 (a) An example panorama represented in spherical projection. (*Continued*)

(b)

(c)

(d)

(e)

(f)

(g)

FIGURE 8.3 (*Continued*) (b–c) Images extracted from the panorama using perspective projection. (d–e) Images extracted from the panorama using perspective projection. (f–g) Images extracted from the panorama using perspective projection. (*Continued*)

(h) (i)

FIGURE 8.3 (*Continued*) (h–i) Images extracted from the panorama using perspective projection.

8.3.2 3D Reconstruction Pipeline

After extraction, the perspective views from the panoramas are processed in an incremental structure-from-motion pipeline, which produces a 3D point cloud from image feature correspondences. This pipeline has four major steps: pair-wise panorama matching, match verification, reconstruction of an initial pair, and incremental addition of the remaining panoramas. If a linear camera path is assumed, without loop closures, panoramas are only matched to their neighbors in the sequence. Otherwise, exhaustive pair-wise matching is employed to test all possible correspondences. The SIFT detector and descriptor is used for feature matching (Lowe, 2004). Matches are verified by finding the essential matrix (Nistér, 2004) relating two panoramas using a progressive sample consensus (PROSAC) loop (Chum and Matas, 2005). After triangulating points using an initial panorama pair, panoramas are incrementally added, more points are triangulated, and bundle adjustment is performed. This is repeated 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.

8.4 SEMIAUTOMATIC GEO-ALIGNMENT

Image-based reconstruction by itself produces a metric reconstruction that is internally consistent. However, this process cannot recover the external orientation of the reconstruction, meaning the direction of gravity, the scale in meters, and the geographic positions of the cameras and 3D points. This external orientation, however, is very useful for many kinds of AR applications. With a geo-aligned reconstruction it is then possible to display geo-referenced information such as map. Other applications which do not display geo-referenced information still benefit from geo-alignment, because it can be used to determine the device's height off the ground and the scale and orientation with which 3D models should be displayed.

To enable these benefits in our AR applications, the semiautomatic geo-alignment procedure described here is employed to determine the external orientation of the reconstruction.

8.4.1 Vertical Alignment

The first step of the alignment procedure is to determine the vertical orientation of the reconstruction. This is determined by two rotation angles that transform the reconstruction to make the negative Z-axis aligned with the direction of gravity. To automatically estimate this alignment, roughly vertical line segments are extracted from all images. The LSD line segment detector (Von Gioi et al., 2010) is applied and all lines with a minimum length of 25% of the image diagonal and an orientation within 45° off-vertical are accepted. Using only roughly vertical lines makes the assumption that the images are taken with a roughly upright orientation, and that there are sufficient upright structures having such lines in the images.

Then, a common vertical vanishing point for all images is determined. Vertical vanishing point hypotheses are generated by repeatedly sampling a pair of lines and finding their intersection point. Each hypothesis is tested against all lines to determine their angular errors with respect to the vanishing point hypothesis. The hypothesis with the greatest number of inliers is selected as the common vertical vanishing point for all images. Figure 8.4 shows an example of vertical lines found to be inliers in one image. After finding the common vertical vanishing point, the rotation which brings this point to vertical $(0, 0, 1)^T$ is determined and applied to the reconstruction.

FIGURE 8.4 Lines on the buildings (in white) are used to determine a common vanishing point and align the vertical axis of the reconstruction with the direction of gravity.

8.4.2 Ground Plane Determination

Once the reconstruction is vertically aligned, the ground plane is determined by considering the Z-coordinate of all 3D points. Assuming that the reconstruction contains many points on the ground, the ground plane should be a peak in the histogram of Z values, near the lower end of the range. Erroneous points in the reconstruction might lie under the ground, so the absolute minimum value should not be used as the ground height. Instead, the height of the ground is initialized to the 80th percentile Z value. The ground height will then be manually tuned by inspecting the reconstruction visually and confirming that the estimated ground plane meets the bottom edges of buildings.

8.4.3 Map Alignment

Now there are four remaining degrees of freedom left: a rotation about the vertical axis, a translation on the X–Y ground plane, and the metric scaling of the reconstruction. An initialization for these remaining transformation parameters is determined manually by visually comparing an overhead, orthographic view of the reconstruction with a map of building outlines from the area, which can be freely downloaded from OpenStreetMap. A simple interactive tool renders the point cloud and building outlines together. The user interactively rotates, translates, and scales the reconstruction until the points roughly match the buildings.

After the user determines a rough initialization, automatic nonlinear optimization is applied to determine the best fit between 3D points and building walls. Each point is assigned to the nearest building wall according to the 2D point-line distance. However, if the point-line distance is greater than 4 m, or the projection of the point onto the line does not lie on the line segment, then the match is discarded. The rotation, translation, and scale parameters are iteratively updated to minimize the point-line distance of all matches, using the Huber loss function for robustness to outliers. The entire optimization procedure is repeated until convergence to find the best point-line assignment and 3D alignment.

An example reconstruction aligned to OpenStreetMap data is shown in Figure 8.5. The panoramas for this reconstruction were captured by walking in a straight line through the center of the Graz Hauptplatz courtyard while holding the Ricoh Theta camera overhead.

8.5 TRACKING THE MODEL

Given a 3D point cloud reconstruction of the scene, the mobile device's position is determined in real time by identifying and localizing feature points observed by the device's camera. 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, the current camera image, and sensor readings, and outputs a pose posterior that estimates the current device position and orientation. The pose prior is provided by the previous tracking iteration. Tracker initialization, and reinitialization after failure, is provided by the procedures discussed in Section 8.6.

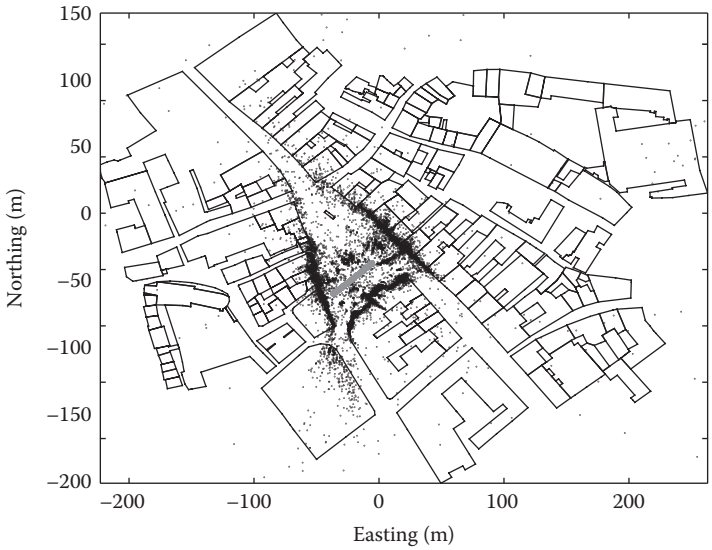

FIGURE 8.5 Point cloud reconstruction aligned to OpenStreetMap building data. The gray dots indicate panorama capture locations, and the black points indicate triangulated 3D points.

8.5.1 Image Representation

We refer the images extracted from the panoramas as keyframes. These keyframes are extended by preparing an image pyramid, meaning that the image is repeatedly half-sampled; the stack of images at progressively smaller resolutions is stored and used to improve patch sampling during tracker operation.

8.5.2 Camera Model

Most mobile phones and tablets have a moderate field-of-view camera with low distortion, thus a simple pinhole camera model without a radial distortion term is sufficient to model it. This model has only one parameter, the focal length, which is determined in a precalibration step. The center of projection is assumed to be the center of the image. The same model is used for the panorama keyframes, which are generated by synthetic warping of the panorama images.

8.5.3 Point Correspondence Search

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 re-projection error.

First, any points that are predicted to lie behind the camera or outside of the image are culled and not considered in further steps of the current tracking iteration.

For each point that passes the culling test, an 8 × 8 pixel template patch is extracted from a keyframe that observes the projected point. This keyframe is called the source, and the camera image is called the target. A perspective warp is used to compensate for the parallax between the source and the target. This perspective warp is determined by the 3D point, X, and its normal, n, and the plane $p = (n_1, n_2, n_3, D)^T$, where $n \cdot X + D = 0$. If the target and source projection matrices are $P = [I \mid 0]$ and $P_i = [R_i \mid t_i]$ respectively, the 3 × 3 perspective warp is

$$W_i = K_{\text{source}} (R_i + t_i v^T) K_{\text{target}}$$

where

$v = -D^{-1} (n_1, n_2, n_3)^T$

K_{source} and K_{target} are the intrinsic calibration matrices of the source and target images, respectively

The determinant of the warp $|W_i|$ gives the amount of scaling between source and target. This warp is computed for all keyframes that observe the point; the keyframe with warp scale closest to one is chosen as the source. This ensures the best resolution when sampling the template patch. The system also chooses the best level of the source image pyramid according to the resolution required to when sampling the template patch.

The template patch is used to search in the target image for the point's current projected location in the image. We search for this location by computing the normalized cross-correlation between the template patch and patches sampled from 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 of 0.7, which was 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.

8.5.4 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 x_i for point X_i. Ten iterations of gradient descent over an M-estimator are used to minimize the re-projection error of all points:

$$e = \sum_i m\left(\left|y_i - x_i\right|^2\right)$$

where

y_i is the projected location of X_i using the current pose estimate

$m(u)$ is the Tukey loss function (Huber, 1981)

The parameters of the Tukey loss function are recomputed to update the weights after each of the first five iterations.

8.5.5 Success Metric

After the pose update, the number N_{found} of points found to be inliers by the M-estimator is counted. This is an indicator of the success of the tracker, that is, whether the pose posterior matches the true pose of the camera. The system requires that at least 100 points have been successfully tracked ($N_{found} \geq 100$).

To ensure an acceptable frame rate, a limit N_{max} is placed on the number of points $N_{attempted}$ that the tracker can attempt to find in a single frame. The system first performs view frustum culling on all points, and then selects $N_{attempted} \leq N_{max}$ points to search for. The point selection is performed by choosing some ordering of visible points and selecting the first N_{max} points from the ordering to be tracked. The question is, which ordering ensures the best tracking performance?

One commonly used approach is to randomly shuffle all the visible points at each frame. However, 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.

A solution to this problem is to randomly order the 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, this sampling procedure reduces pose jitter in comparison to sampling a new random ordering of points at each frame.

8.5.6 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. 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.

To address this problem, live keyframe sampling is incorporated into the tracking system. During tracker operation, keyframes are collected from the current video stream and added to the set of images used for patch projection. The tracker preferentially projects 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 that 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 m, a new keyframe is sampled, and the inlier measurements are associated to the corresponding 3D points to be used for future patch projection.

8.6 TRACKER INITIALIZATION

The patch tracking method described in Section 8.5 is fast enough for real-time operation, but requires an initial position to start the iterative tracking procedure. The region of convergence for the tracker is too small to make it feasible to use the GPS and compass reading as an initialization (see Section 8.7.2). Instead, visual localization procedures are employed that are capable of determining the camera pose within a wide range of possible positions.

The task of visual localization is challenging in the outdoor case. This is because the range of possible views is large compared to an indoor setting. When tracking a desk, for example, the camera can reasonably be expected to move within a small range of distances from the desk surface. In a building courtyard or street-side setting, however, the camera could be far from the original point of capture, but the building would still be visible because of its size. This mandates localization strategies that are robust to large changes in perspective and scale.

This section presents two different methods for tracker initialization and reinitialization after tracking failure. The image-based method is fast enough to be computed in real time but is limited in range. The feature-based method offers a more robust solution to the visual localization problem, but requires significant computation as well as storage for the descriptor database. The way these methods are combined is explained in the system design overview given in Section 8.7.

8.6.1 Image-Based Method

The image-based localization method is relatively simple and is easily implemented. A cache of recently seen images is stored along with their known poses. When the tracking system fails and enters the *lost* state, the system matches the current image to the cache to find pose hypotheses. The best matching image is used as a pose prior to start the tracker at the next frame.

The image cache is generated during tracker operation and can be saved for reuse in future tracking sessions. During tracker operation, a tracked image is added to the cache when tracking is successful and the closest keyframe in the cache is more than 1 m different in position or 45° different in orientation.

To find image matches, a variant of the small blurry image (SBI) matching procedure is used (Klein and Murray, 2008). Images in the cache are down-sampled to the fifth image pyramid level (meaning that they are half-sampled four times). This same down-sampling is applied to the current query image from the camera. Then each cache image is compared to the query image using the normalized cross-correlation score (Gonzalez and Wood, 2007). The pose of the image with the highest correlation score is used as the initialization for the tracker in the next frame.

The SBI localization method is suitably fast even for a large number of cache images. However, it requires the query camera to be relatively close to a cache image. Beyond a small amount of translation or rotation, the query frame will not match to any cache image. Thus, this method is impractical for outdoor localization in a large space, unless a very dense coverage of cache images is acquired.

8.6.2 Feature-Based Method

Alternatively, feature matching can be used to extend the range of poses where localization can be achieved. Given a query image, the system extracts features and searches for correspondences in the panorama keyframe database. Then, a robust sampling procedure is used to find a subset of inlier correspondences that support a common pose estimate.

Each feature from the query image is matched to its nearest neighbor in the set of all features in the database according to the Euclidean distance between SIFT descriptors. Approximate nearest-neighbor search is performed using a kd-tree for speed (Lowe, 2004). Then the camera pose is robustly estimated using the PROSAC procedure (Chum and Matas, 2005) and the three-point absolute pose algorithm (Fischler and Bolles, 1981).

An alternative approach is to apply a document retrieval technique (Sivic and Zisserman, 2003). A vocabulary tree (Nistér and Stewenius, 2006) is used to hierarchically organize the descriptors so that each descriptor is identified by its cluster (or *word*). Given a query frame, standard *tf-idf* weighted document matching is applied to order the keyframes by similarity (Sivic and Zisserman, 2003). The top K documents are then subjected to geometric pose verification to find a suitable match. For each top-ranked document, the nearest-neighbor matching and pose estimation procedure described earlier is performed, using only features from the single image that was retrieved. This image retrieval approach scales with database size better than the performing nearest-neighbor matching with entire database. However, there must exist in the database a single view that has enough visual overlap with the query for the procedure to work. Irschara et al. developed a method to increase the set of views in the database synthetically, which increases the range of the image retrieval technique (Irschara et al., 2009).

8.7 SERVER/CLIENT SYSTEM DESIGN

This section gives an explanation of how the various components described earlier are organized into a complete outdoor tracking system for AR applications.

8.7.1 Server/Client System Overview

The overall system design is illustrated in Figure 8.6. The online tracking and image-based reinitialization components are computed directly on the mobile client device, as these are relatively lightweight operations that can be computed in real time on such restricted hardware. The feature-based localization component is computed on a remote server or computing cloud, where storage and computation are essentially unrestricted. The system's preparation and its online operation procedure are described in more detail in the following.

First, the omnidirectional video is processed in the offline reconstruction process to produce the 3D point cloud model with panorama keyframes and feature descriptors. The point cloud and the keyframes are copied to the mobile client device; the descriptors are not needed for tracking and thus do not need to be copied onto the client device.

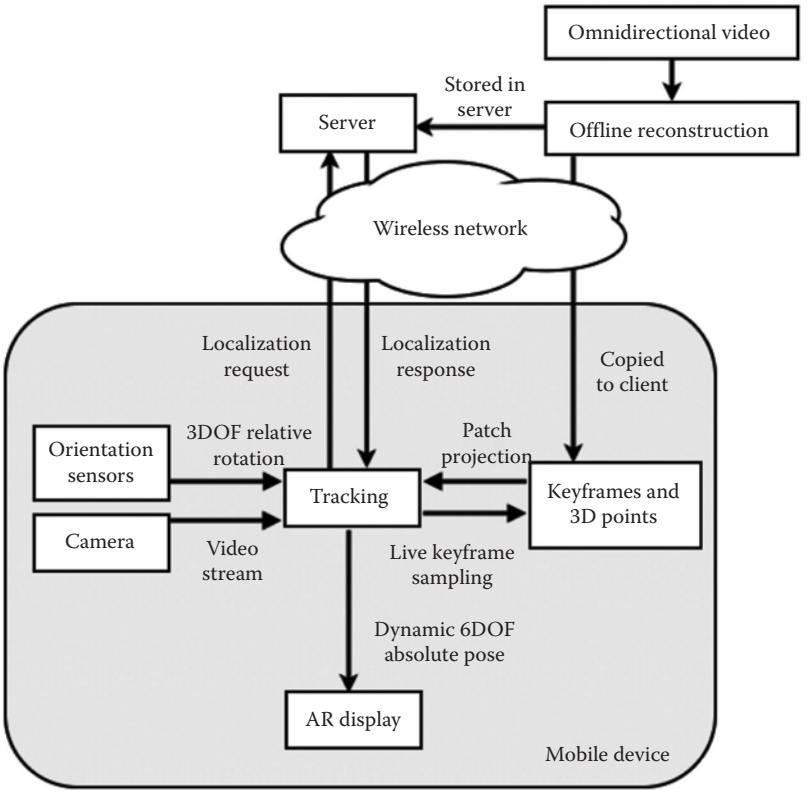

FIGURE 8.6 Tracking system overview with server/client design.

The tracking system runs in real time on the client device in the following loop. First, the system tries to track the model using the previous pose estimate. The incremental rotation estimate provided by the inertial sensors in the device is preapplied to the previous pose estimate to compensate for fast motion. If tracking fails, then the image cache is searched using the current camera image. Tracking is then tried again using the pose prior provided by the best match from the image cache. If this fails, then the system generates a localization query that is sent to the server over the wireless network. While the server processes the query, the system continues attempting to restart tracking using the image cache. When the query response is received, the computed pose is used to restart tracking.

8.7.2 Latency Analysis

Due to network communication time and server computation time, feature-based localization introduces latency between an image query and a pose response. During this time, the camera might be moved from its query position, introducing error in the localization pose estimate. Thus, the system needs some ability to handle an outdated pose estimate from the localization system.

The region of convergence of the tracker determines the amount of error in the pose prior that the system can tolerate. 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 that 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 q_{err} degrees will cause a pixel offset of x_{err} pixels:

$$x_{\text{err}} = f \tan(q_{\text{err}})$$

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 $q_{\max}$.

The system uses an effective search radius of $4 \cdot 2^3 = 32$ pixels, and the Apple iPad 2 camera used for testing has a focal length of $f = 1179.90$. Thus, the maximum rotational pose error is $q_{\max} = 1.55°$. This limit could be a problem if localization latency is 1 s or more.

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

$$x_{\text{err}} = f t_X Z$$

For the iPad 2 camera, the maximum translation is $t_X/Z = 0.03$. Given a building that is 12 m away, the maximum translation would be about 1/3 m. This as well would be a limitation for localization, given the distance a fast-walking user could cover in 1 s.

This analysis suggests in general that the complete time for the localization query to be sent, processed, and returned—the localization latency—should be within 1 s. Timing data from our experiments is given in Section 8.1.

8.7.3 Sensor Integration

To overcome the problem of rotational movement during the localization latency period, the inertial sensors in the device are used to maintain an estimate of rotational movement. The estimated difference in rotation between the localization query and response is preapplied to the localization response before attempting to initialize the tracker.

A similar approach could be applied to estimate translational movement based on accelerometer readings. However, the accelerometer found in typical consumer devices, such as the iPad 2, are too noisy to be used for estimating translation, even over a brief period. Fortunately, translational error during the latency period is not an issue in larger environments such as typical urban scenes. This is because generally the distance to the buildings is such that small translational movements do not cause significant parallax in the image.

8.8 EVALUATION

This section reports on evaluations of several aspects of the system and shows that it provides sufficient tracking performance to support many kinds of geo-referenced mobile AR applications.

8.8.1 Speed

Localization queries are processed on a remote server while the mobile tracker continues running. This means that the server does not have to respond in real time, since the processing happens in the background. However, the processing time should be as short as possible to provide a smooth user experience, and ideally within 1 s, as determined in Section 8.7.2.

Average timings were recorded using an Apple Mac Pro with a 2.26 GHz Quad-Core Intel Xeon and 8 GB RAM. The model tested has 21 panoramas, 3691 points, and 6823 features. Most of the computation time is spent on SIFT feature extraction (900 ms) and PROSAC pose estimation (500 ms). 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, the average localization latency is about one and a half seconds. In practice we have experienced localization times of 2–3 s for a larger model. However, the processing speed could be greatly improved by using GPU implementations of the feature extraction and pose estimation steps.

The speed of online tracking on the client device was evaluated using an Apple iPad 2 tablet. Feature-based tracking on the mobile device consists of three steps that constitute the majority of computation time per frame: point culling (0.005 ms per point); patch warp (0.02 ms per point); and patch search (0.033 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:

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

With multithreading on the dual-core iPad 2, the processing time is approximately reduced by half. For a model with 3691 points, 1024 tracked points, and 4 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, and frame rates of 15–20 fps tracking are achievable.

8.8.2 Accuracy Tests with Differential GPS

To test the absolute positional accuracy possible with the system, a differential GPS receiver was attached to the iPad 2. Differential GPS receivers use measurements from GPS satellites as well as a correction signal from a nearby base station in order

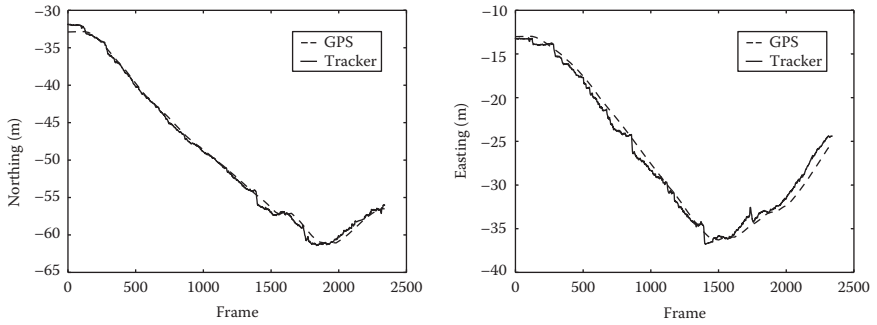

FIGURE 8.7 Comparison of the camera position estimates from the visual tracking system with ground truth position estimates from the differential GPS receiver.

to attain ground truth positional estimates with accuracy under 10 cm. Because the GPS receiver produces positional readings at a rate of 1 Hz, linear interpolation was used to up-sample the signal to 30 Hz.

A test video with the differential GPS receiver was recorded in the Graz Hauptplatz while observing the Rathaus (City Hall). The panoramic reconstruction of this area was made from 37 panoramas taken with the Ricoh Theta camera. The resulting reconstruction contains 14,523 points. The semiautomatic alignment method described in this chapter was used to georegister the model with respect to building outlines from OpenStreetMap. An overhead view of the point cloud is shown in Figure 8.5.

A comparison of the differential GPS track and the positional track created with our system is shown in Figure 8.7. The system achieved an average error of 0.72 m in the easting direction and 0.38 m in the northing direction. This shows that our system provides better accuracy than consumer GPS, which has an accuracy of about 3 m with a high-quality receiver.

8.8.3 Augmentation Examples

Several prototypes have been developed and tested to evaluate the use of our modeling and tracking system for AR applications. Example screen captures from these prototypes are shown in Figure 8.8.

The first prototype is a landscape design application. In a large courtyard on the UC Santa Barbara campus, the user can place virtual trees on the large grassy area between the buildings. As trees are placed, the user can move around to view how the trees would look from different angles.

A second prototype tests the use of video game graphics. In this application, a landing spaceship is rendered into another building courtyard on the UCSB campus at the spot on the ground where the user touches the screen. Using an assumed position of the sun, accurate shading and shadows are rendered, to increase the realism of the rendering.

A third prototype was created to test architectural rendering. Here, a reconstruction of a city street (Branch Street in Arroyo Grande, CA) was created by holding the panorama camera out on the sunroof of a car and driving down the street to capture

(a)

(b)

(c)

FIGURE 8.8 Example images of the tracking system in use with 3D models rendered over the camera image. (a) Synthetic trees planted in the grass. (b) A spaceship landing in the courtyard, rendered with lighting and shadow effects. (c) Virtual lamps affixed to the side of the building.

the buildings on either side. Then, a user standing on the sidewalk can add architectural elements such as virtual lamps to the building facades by simply touching the screen at the points on the wall where they should be placed.

8.9 DISCUSSION

From these evaluations, it can be concluded that visual modeling and tracking offers a compelling solution to device pose estimation for mobile AR applications. The approach enables high-accuracy tracking at real-time rates with consumer hardware. Experience with the prototype applications suggests that the pose estimation is of sufficient quality to make objects appear to *stick* to surfaces, such that they seem truly attached to a wall or a ground. Using simple rendering techniques such as shading and shadowing also helps to improve the perceived realism of the rendered graphics.

The major limitation of this approach is that the system is generally restricted to operation from viewpoints where the scene is visually distinctive and able to be recognized by its appearance. For many viewpoints, this is not the case, such as texture-less building walls, and the sky or the ground. In addition, many scenes contain repetitive textures, such as grids of windows, that confuse the visual localization system and lead to system failure. One possible solution to these problems would be to further integrate other position and motion sensors, such as a GPS receiver, accelerometer, gyroscope, and compass, to complement the visual tracker.

The source code for the system described in this chapter is publicly available for download, testing, and further development at http://www.jventura.net/code.

8.10 FURTHER READING

In this final section, references are provided so that the interested reader can find more details about this work, as well as canonical references to learn more about this research area and other approaches to the problem. This reference list is not intended to be exhaustive, but is instead a starting point for further investigation.

More details about the system described in this chapter can be found in our research papers (Ventura and Hollerer, 2011, 2012a,b) and Ventura's doctoral dissertation (Ventura, 2012). The 3D reconstruction pipeline is based on that of Snavely (2008), Snavely et al. (2006), with modifications to handle cameras arranged in a panoramic rig. The fundamentals of multiview geometry are discussed extensively in the essential textbook by Hartley and Zisserman (2004).

One classic reference for camera pose estimation is that of Fischler and Bolles, who introduced a solution to the camera pose estimation problem as well as the Random Sample Consensus (RANSAC) method for finding a consistent set of observations from noisy data (Fischler and Bolles, 1981). PROSAC is a more efficient variant of RANSAC and is applied in this work (Chum and Matas, 2005). Most modern methods rely on the SIFT method to detect feature points and kd-trees for approximate nearest-neighbor feature matching (Lowe, 2004). Many researchers have also investigated more scalable approaches to feature matching (Arth et al., 2009; Li et al., 2010; Sattler et al., 2011). The document-based approach to image retrieval was introduced by Sivic and Zisserman (2003) and expanded by others to include vocabulary trees (Nistér and Stewenius, 2006), geometric verification (Philbin et al., 2007), and virtual images (Irschara et al., 2009).

Camera-based tracking also has a long history of research. In the AR context, one canonical work is by Lowe who used SIFT descriptors for initialization and tracking (Skrypnyk and Lowe, 2004). More recently, the landmark work of Klein and Murray introduced Parallel Tracking and Mapping (PTAM), where points are triangulated and tracked simultaneously in an efficient manner (Klein and Murray, 2007). The tracking method described in this chapter is adapted from this work. Alternatives to the point-based approach are possible, such as using a wireframe (Klein and Murray, 2006) or textured 3D model (Reitmayr and Drummond, 2006). Researchers in AR systems have also considered approaches to outdoor pose tracking that use the camera in combination with other dedicated position and velocity sensors (Oskiper et al., 2012).

REFERENCES

Arth, C., Wagner, D., Klopschitz, M., Irschara, A., and Schmalstieg, D. (2009). Wide area localization on mobile phones. In *ISMAR'09 Proceedings of the 2009 Eighth IEEE International Symposium on Mixed and Augmented Reality* (pp. 73–82). Washington, DC: IEEE Computer Society.

Chum, O. and Matas, J. (2005). Matching with PROSAC-progressive sample consensus. In *CVPR 2005. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005* (Vol. 1, pp. 220–226). Washington, DC: IEEE Computer Society.

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

Gonzalez, R. and Wood, R. E. (2007). *Digital Image Processing*. Upper Saddle River, NJ: Pearson/Prentice Hall.

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press.

Huber, P. J. (1981). *Robust Statistics*. New York: John Wiley & Sons.

Irschara, A., Zach, C., Frahm, J. M., and Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *CVPR 2009. IEEE Conference on Computer Vision and Pattern Recognition, 2009* (pp. 2599–2606). Washington, DC: IEEE Computer Society.

Klein, G. and Murray, D. (2006). Full-3d edge tracking with a particle filter. In *British Machine Vision Conference (BMVC'06)*. Manchester, U.K.: British Machine Vision Association.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *ISMAR'07: Proceedings of the 2007 Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality* (pp. 225–234). Washington, DC: IEEE Computer Society.

Klein, G. and Murray, D. (2008). Improving the agility of keyframe-based SLAM. In *ECCV'08: Proceedings of the 10th European Conference on Computer Vision: Part II* (Vol. 5303 LNCS, pp. 802–815). Berlin, Germany: Springer-Verlag.

Li, Y., Snavely, N., and Huttenlocher, D. (2010). Location recognition using prioritized feature matching. In *ECCV'10: Proceedings of the 11th European Conference on Computer vision: Part II* (pp. 791–804). Berlin, Germany: Springer-Verlag.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(6), 756–777.

Nistér, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR'06 Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vol. 2, pp. 2161–2168). Washington DC: IEEE Computer Society.

Oskiper, T., Samarasekera, S., and Kumar, R. (2012). Multi-sensor navigation algorithm using monocular camera, IMU and GPS for large scale augmented reality. In *ISMAR'12: Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (pp. 71–80). Washington, DC: IEEE Computer Society.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *CVPR'07. IEEE Conference on Computer Vision and Pattern Recognition, 2007* (pp. 1–8). Washington, DC: IEEE Computer Society.

Reitmayr, G. and Drummond, T. W. (2006). Going out: Robust model-based tracking for outdoor augmented reality. In *ISMAR'06: Proceedings of the Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality* (pp. 109–118). Washington, DC: IEEE Computer Society.

Sattler, T., Leibe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2D-to-3D matching. In *ICCV'11 Proceedings of the 2011 International Conference on Computer Vision* (Vol. 43). Washington, DC: IEEE Computer Society.

Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *ICCV'03 Proceedings of the Ninth IEEE International Conference on Computer Vision* (Vol. 2, pp. 1470–1477). Washington, DC: IEEE Computer Society.

Skrypnyk, I. and Lowe, D. G. (2004). Scene modelling, recognition and tracking with invariant image features. In *ISMAR'04 Proceedings of the Third IEEE/ACM International Symposium on Mixed and Augmented Reality* (pp. 110–119). Washington, DC: IEEE Computer Society.

Snavely, K. (2008). Scene reconstruction and visualization from Internet photo collections. Dissertation, University of Washington, Seattle, WA.

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics (TOG)—Proceedings of ACM SIGGRAPH 2006*, *25*(3), 835–846.

Ventura, J. (2012). *Wide-Area Visual Modeling and Tracking for Mobile Augmented Reality* (T. Hollerer, Ed.). Santa Barbara, CA: University of California.

Ventura, J. and Hollerer, T. (2011). Outdoor mobile localization from panoramic imagery. In *ISMAR'11 Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality* (pp. 247–248). Washington, DC: IEEE Computer Society.

Ventura, J. and Hollerer, T. (2012a). Structure from motion in urban environments using upright Panoramas. *Virtual Reality*, *17*(2), 147–156.

Ventura, J. and Hollerer, T. (2012b). Wide-area scene mapping for mobile visual tracking. In *ISMAR'12 Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (pp. 3–12). Washington, DC: IEEE Computer Society.

Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. (2010). LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(4), 722–732.