Going out: Robust Model-based Tracking for Outdoor Augmented Reality

Gerhard Reitmayr * Tom W. Drummond †

Engineering Department
Cambridge University
Cambridge, UK

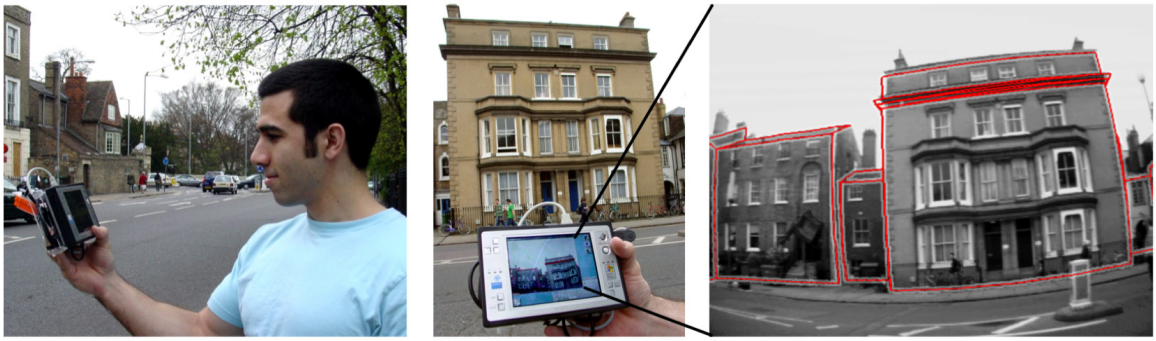

Figure 1: (Left) A user operating a handheld augmented reality unit tracked in an urban environment. (Middle) Live shot showing the unit tracking a building. (Right) Screenshot from a pose close to the left images with overlaid building outline.

ABSTRACT

This paper presents a model-based hybrid tracking system for outdoor augmented reality in urban environments enabling accurate, realtime overlays for a handheld device. The system combines several well-known approaches to provide a robust experience that surpasses each of the individual components alone: an edge-based tracker for accurate localisation, gyroscope measurements to deal with fast motions, measurements of gravity and magnetic field to avoid drift, and a back store of reference frames with online frame selection to re-initialise automatically after dynamic occlusions or failures. A novel edge-based tracker dispenses with the conventional edge model, and uses instead a coarse, but *textured*, 3D model. This yields several advantages: scale-based detail culling is automatic, appearance-based edge signatures can be used to improve matching and the models needed are more commonly available. The accuracy and robustness of the resulting system is demonstrated with comparisons to map-based ground truth data.

CR Categories: H.5.1 [Information Systems]: Multimedia Information Systems—Augmented Reality I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking, Sensor Fusion

1 INTRODUCTION

Augmented reality (AR) is a promising user interface technique for mobile, wearable computing and location-based systems. The goal of presenting information in situ has triggered a large body of research into mobile AR systems and interfaces. However, accurate and robust localisation is still elusive, which dictates the usually experimental nature of current systems.

*e-mail:gr281@cam.ac.uk
†e-mail:twd20@cam.ac.uk

Traditionally, AR systems developed for outdoor use rely on GPS for position measurements and magnetic compasses and inertial sensors for orientation. While GPS has satisfactory accuracy and performance in open spaces, its quality deteriorates significantly in urban environments. Both the accuracy and the availability of GPS position estimates are reduced by shadowing from buildings and signal reflections. Similarly, inertial sensors are prone to drift and magnetic sensors are disturbed by local magnetic fields encountered in urban environments. A computer vision-based localisation component can provide both accurate localisation and robustness against these environmental influences.

Model-based tracking approaches appear to be the most promising among the standard vision techniques currently applied in AR applications. While marker-based approaches such as ARToolkit [14] or commercial tracking systems such as ART provide a robust and stable solution for controlled environments, it is not feasible to equip a larger outdoor space with fiducial markers. Hence, any such system has to rely on models of natural features such as architectural lines or feature points extracted from reference images.

The choice of models is usually dictated by the detector used in a model-based tracking system. Point based systems use databases of 3D point locations and their visual descriptors while edge-based systems employ a CAD model of salient edges and faces to compute occlusions. These approaches work well for small scale applications such as tracking individual parts or small indoor environments. Creating such detailed models for large environments becomes a daunting task, and the geometric complexity poses new challenges to the rendering component required to determine visibility and appearance of edges and points. For example, dense clustering of line features at a larger distance requires appropriate sampling to determine when lines merge and become indistinguishable by the detector.

By contrast, the approach presented here does not use a detailed edge model of the environment, but instead employs a *textured* 3D model. The edges to be tracked are then dynamically determined at runtime by performing edge detection on a *rendering* of the 3D model from the current pose prediction. This provides three signif-

icant advantages: The system automatically performs detail culling and only searches for edge features that are likely to be visible at the current scale. The system is able to extract an appearance signature for each edgel that it wishes to localise in the live feed, thus increasing the accuracy of matching. Finally, textured 3D models are more commonly available and commercial tools already exist to create them, because domains such as geographical information systems and computer graphics are increasingly interested in them.

Outdoor environments also challenge the robustness of a tracking system to occlusions, large variations in lighting and generally bad viewing conditions. A convincing tracking solution must overcome inherent limitations of individual techniques by combining different complementary methods. This work takes the common approach of combining edge-based trackers with inertial sensors to provide more accurate priors under fast motions. Still, occlusions may provide wrong motion cues due to poor data association and generate failures in a visual tracking system. For, example a large truck driving through the field of view can induce a motion in the system's filter because the detector makes false correspondences with features of the vehicle. A robust system should detect and recover from such disturbances.

The edge-tracker using 3D building models is described in section 3 with details on the rendering, edge extraction and matching steps described in section 3.2. Additional sensor fusion with inertial and magnetometer sensors provide robustness under fast motions (see section 3.3). A point-based image matching component implements a fallback method to recover from failures of the edge-tracking (see section 4). The system is demonstrated in a simple location-based game (see section 5) and evaluated against ground truth based on map data of the City of Cambridge (see section 6).

2 Related work

The first examples of outdoor mobile augmented reality typically relied on GPS for localisation and inertial sensors coupled with magnetic compasses to provide orientation [1]. Examples of such systems include Columbia's Touring machine and MARS [8, 10], the Battlefield Augmented Reality System [2], work by Thomas et al. [28] and the later Tinmith System by Piekarski [19].

To improve accuracy over pure GPS and compass-based systems, You et al. [33] describe a hybrid tracking system for outdoor AR combining optical tracking and inertial sensors. The TOWNWEAR system [24] uses a high precision laser gyroscope to achieve drift-free registration for a mobile AR system.

Vision based localisation in urban environments has been studied before. Johansson & Cipolla [13] investigated pose estimation from a single image in a city scene based on architectural patterns such as grids of windows and doors. This work was extended by Robertson & Cipolla [22] by matching a single image into a database of views with known positions. Such a system provides a very general approach to the problem of localisation without any prior knowledge, but does not solve accurate localisation in realtime.

To operate in realtime, computationally less expensive approaches are required. Imaged-based methods reduce the number of reference images by employing panoramic images. Stricker [27] describes a system that matches video images into a reference panoramic image using Fourier transform-based techniques. Similarly, Lee et al. [17] use an omni-directional camera to track between a small set of panoramic reference images.

Using line features has been proposed as well. Coors et al. [6] describe a localisation system based on a 3D city model where visible edges are globally matched to edges extracted from a video image. However, they do not present results on the operation of the final system. Behringer [4] uses an edge-based method to align the horizon silhouette extracted from elevation grid data with a video image for estimating orientation of a mobile AR system.

Another approach is the use of point features. Skrypnyk & Lowe [26] describe a classic point-based system for natural feature tracking using a database of highly discriminate feature descriptors. Simon et al. [25] describe a point-based pose tracker that requires no further model and is demonstrated with outdoor scenes. Ribo et al. [20] introduce a hybrid tracking system for outdoor augmented reality using a database of feature points established offline from a 3D model and a set of reference images. Gyroscopes are used to increase the robustness of the system. Lepetit et al. [18] present a point-based tracking system that uses a 3D model to reproject 2D point correspondences between video frames to track camera motion. Additionally, they also address initialisation using a set of registered reference images to extract a set of feature points. A startup a video frame is matched against this database to compute an initial pose. Our point-based recovery method was inspired by these systems. However, our system automatically selects reference frames online to free the user from the additional overhead of selecting and creating such frames.

Vacchetti et al. [29] describe a combination of edge-based and point-based tracking for AR. Registered reference images provide feature points that are rendered from the current pose prediction to add an edge-based tracker. They note the problem of mismatching within dense clusters of lines which is dealt with using a robust optimisation technique. Similarly, Rosten & Drummond [23] describe a combination of edge and point tracking to increase robustness under fast camera motions.

The edge-tracking algorithm used in this work was described by Drummond & Cipolla [7] and was already successfully applied to augmented reality in the work of Klein & Drummond [15, 16]. The latter also includes sensor fusion with inertial sensors to improve the edge localisation. The principal difference to these former works is the use of a texture-based model and online extraction of edge features for matching to the input image. Wuest et al. [32] also use an appearance-based approach to edge detection by learning gaussian mixture models of the colours on each side of an edge.

Jiang et al. [12] propose a similar system for outdoor tracking that integrates line-based tracking with gyroscope measurements to accommodate fast rotations. Lines are defined offline in a CAD model and matched using a heuristic line selection method to avoid wrong line correspondences. Our system uses a simpler textured model and an appearance-based edge detection method to improve matching. It performs as well in similar validation situations and is less computationally expensive, yielding a higher frame rate.

Creating large scale models of urban environments is the focus of a number of research groups. For example, see Hu et al. [11] for an extensive survey over techniques used in creating 3D city models. Fruh & Zakhor [9] describe a highly automated method allowing fast creation of complex models. A simplified version of the resulting models would provide a good basis for our tracking system.

3 Tracking framework

The overall tracking framework consists of an edge-based tracking system combined with a sensor for inertial and magnetic field measurements. A Kalman filter with a constant velocity model is employed to fuse measurements from both components.

3.1 Edge tracking

The edge-based tracking system tracks the pose of a camera mounted to a handheld AR system in an urban outdoor environment. This technique has previously been applied to HMD-based AR [15] and a tablet computer-based system [16] by Klein & Drummond. Therefore, we give only a short review of the system's operation here.

(a) (b) (c) (d) (e)

Figure 2: Overview of the tracking process. A view is rendered from the prior camera pose (a), then edgels are extracted from the grayscale image (b). The prior pose overlaid over the video image (c) is updated with measurements (d) to yield the posterior pose (e).

The tracking system relies on a 3D model of the scene to be tracked. In former systems the 3D model describes salient edges and occluding faces. Using a prior estimate of camera pose, this 3D model is projected into the camera's view for every frame, computing the visible parts of edges. A point $\boldsymbol{x}_w = (x_w, y_w, z_w, 1)$ in world coordinates is projected into the view to the point (u, v) using the camera pose T_{cw}^- as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \text{CamProj}(T_{cw}^- \boldsymbol{x}_w). \tag{1}$$

T_{cw}^- is a 4×4 rigid transformation matrix, consisting of rotation and translation, mapping points from the world coordinate system w into the camera coordinate system c. Such matrices form a representation of the Lie group $SE(3)$. They can be parametrised with a six-vector μ corresponding to translations along and rotations around the three axes using the exponential map of the Lie group $SE(3)$. We use $^-$ to denote prior poses for estimation and filtering.

The function CamProj$(\cdot)$ models the projection from camera frame to image coordinates as

$$\text{CamProj}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} f_u & 0 & p_u \\ 0 & f_v & p_v \end{pmatrix} \begin{pmatrix} r' \frac{x}{z} \\ r' \frac{y}{z} \\ 1 \end{pmatrix} \tag{2}$$

where $r = \sqrt{(\frac{x}{z})^2 + (\frac{y}{z})^2}$, $r' = 1 + \alpha r^2 + \beta r^4$ to compensate for radial lens distortion.

The projection of the 3D model will not correspond to the edges in the video frame because of camera motion. The goal is then to compute the motion M of the camera required to align the projection with the video image to provide a posterior pose estimate

$$T_{cw} = M T_{cw}^-. \tag{3}$$

The edge-based tracking system calculates the motion M from measurements of edge displacements along the edge normal. Sample points are chosen along visible edges and searches in the direction of the edge normal for the nearest edgel in the video image are performed. For every sample point i, the distance d_i to the closest edgel is computed and stored.

The motion M is parametrised by the six-vector μ using the exponential map of the Lie group $SE(3)$. Equation 1 is differentiated with respect to these six motion parameters to provide a $N \times 6$ Jacobian matrix J such that

$$J_{i,j} = \frac{\partial d_i}{\partial \mu_j} = n_i \begin{pmatrix} \frac{\partial u}{\partial \mu_j} \\ \frac{\partial v}{\partial \mu_j} \end{pmatrix} \tag{4}$$

where n_i is the unit-length edge normal for the sample point i in the image. The motion parameters can be found by solving the equation

$$J\mu = \boldsymbol{d} \tag{5}$$

where $\boldsymbol{d}$ is the N-dimensional vector of distance measurements d_i. For standard least-squares optimisation the solution takes the form $\mu = (J^T J)^{-1} J^T \boldsymbol{d}$. In practice, a robust M-estimator is used.

A model of the errors in the measurement of camera pose is required to incorporate the new pose into a Kalman filter for sensor fusion as described in section 3.3. Errors are modelled as small disturbances parametrised by a 6-vector ε that is normally distributed with mean zero

$$\varepsilon \sim N(0, C). \tag{6}$$

The noisy measurement $\hat{M}$ is then given as

$$\hat{M} = \exp(\varepsilon) M. \tag{7}$$

The covariance C of the measurement can be calculated using the Jacobian J and measurement noise vector $\boldsymbol{\sigma}$ as follows

$$C = (J^T \operatorname{diag}(\boldsymbol{\sigma})^{-1} J)^{-1}. \tag{8}$$

3.2 Samples from a textured 3D model

The original version of the edge-based tracking system used a CAD model of salient edges of the scene and computed visible edges and sample points along these edges. Here, we introduce a novel approach to computing the visible edges including their appearance and its application to edge-tracking.

Using a coarse, but *textured* 3D model of an urban environment, a more realistic view from the prior camera pose is rendered, from which edgels are extracted using a standard edge detector. These edgels are then projected back onto the model to obtain the 3D coordinates of the sample points. The edge search is conducted using an appearance-based model of the edge instead of a simple edge detector. See Figure 2 for an example of the individual steps involved.

There are a number of advantages to using a textured 3D model instead of a pure edge model. The complexity of the geometric primitives is much lower for a low-count polygon model than for a detailed line model capturing all the regular structure appearing in urban environments such as windows, doors and other facade features. However, such details are necessary to provide tracking in a wide range of poses and improve the robustness by providing more measurement points. Modern video card hardware offers adequate support for texture rendering making such an approach feasible. Also, rendering polygonal models takes care of occlusions automatically and avoids a more complex hidden-line rendering algorithm to determine the visible parts of the edge model. Similarly, [32] uses OpenGL-based rendering to determine visibility.

More importantly, the texture based rendering provides automatic filtering of the appearance model based on the scale of features in the image. A pure line-based renderer would create dense clusters of edges for facades containing a lot of surface details which cannot be distinguished in the video image. This leads to wrong data associations and incorrect pose estimates. These problems are well known for edge-based trackers as discussed in [29, 23]. Because we select edgels from a filtered view of the

model, denser clusters of edges are more likely to merge visually and become a single edge to be used in tracking.

The system renders a view of the textured 3D city model as seen from the prior pose of the camera using OpenGL and a scene graph library. The OpenGL projection matrix is set to the linear part of the camera model to obtain an image that is of similar scale as the video image. Both the frame buffer and the depth buffer are then read back into memory for further processing. To reduce time spent reading back the depth buffer, the system determines a subset of pixels needed based on the extracted edgels locations (as described in the next step) and trades off the number of calls and area of rectangles with a simple greedy algorithm to minimise total readback time. This approach saves 30% – 50% of the time required to read the whole depth buffer.

Next, the frame buffer image is converted to grayscale and a Canny edge detector [5] is applied to find edgels in the image: after convolution with a gaussian ($\sigma = 1.7$), the gradient image is computed. Edgels with gradient magnitude above a fixed threshold are selected and subjected to non-maximum suppression. The resulting edgels are returned as individual edge points. Finally, a randomly-selected subset of fixed size is retained for further processing.

For each edgel, the corresponding 3D location is computed by reversing the rendering transformation implemented by OpenGL. The depth value of the edgel is read from the depth buffer. To select the right depth at edges between the model and the background or different parts of the model, the pixels along the edge normal are checked for closer depth values. If such a pixel is found it is used instead. Finally, the pixel location and depth value are transformed to OpenGL normalised device coordinates and further transformed by the inverse of the projection matrix to camera coordinates. The resulting point is the visible 3D point of the model which corresponds to the edgel in the view. Note that the 3D points are already in camera coordinates. Therefore, the prior pose for the optimisation in equation (1) is the identity.

The frame buffer image helps to localise edges more reliably using an appearance-based approach. Instead of searching for a step in image intensity along the edge normal, the tracker compares a 1D window (e.g. size 11 pixels) centred around the edgel in the frame buffer image with a corresponding window in the video image. This window slides along the edge normal in the video image and the normalised cross-correlation (NCC) score between the two windows is computed. The location of the window with the maximum score is the corresponding location and the signed distance is the required displacement measurement. In contrast to [32], the use of normalised intensity values provides already an invariant signature that reduces the likelihood of wrong data association. However, learning about appearance changes in the environment would require an update of the model's texture.

3.3 Inertial measurements and sensor fusion

An additional sensor pack provides gyroscopic measurements of rotational velocity, a 3D acceleration vector, and a 3D magnetic field vector. These measurements are combined with the pose estimates of the optical tracking component to make the system more robust in the presence of fast motions. Sensor fusion is accomplished using an extended Kalman filter (EKF) similar to the SCAAT approach [31] described by Welch & Bishop. Again, the implementation follows the description in [16] of which a brief review is given here.

The filter employs a constant velocity model with 12 parameters: 6 parameters for camera pose and 6 parameters for velocity. The pose is stored as the transformation matrix T_{cw} while the velocity is stored as a 6-vector $\boldsymbol{v}$ in the camera coordinate frame c. The state estimate at time t is

$$\hat{x}_t = \{\hat{T}_{cw,t}, \hat{\boldsymbol{v}}_t\} \tag{9}$$

and relates to the real state as

$$\begin{aligned} \hat{T}_{cw,t} &= \exp(\varepsilon_{\text{pose}}) T_{cw,t} \\ \hat{\boldsymbol{v}}_t &= \varepsilon_{\text{vel}} + \boldsymbol{v}_t \end{aligned} . \tag{10}$$

Therefore the state error is the 12-vector $\varepsilon_t = (\varepsilon_{\text{pose}}, \varepsilon_{\text{vel}})$ and is modelled as normally distributed

$$\varepsilon_t \sim N(0, P_t) \tag{11}$$

where P_t is the filter's state error covariance at time t.

A state update for a time step δ is computed using the exponential map

$$T_{t+\delta} = \exp(\delta \boldsymbol{v}_t) T_t, \quad \boldsymbol{v}_{t+\delta} = \boldsymbol{v}_t. \tag{12}$$

The covariance matrix P_t of the filter state is updated according to the dynamic model as

$$P_{t+\delta} = A(\delta) P_t A^T(\delta) + \sigma^2 Q(\delta), \tag{13}$$

$$A(\delta) = \begin{pmatrix} I_6 & \delta I_6 \\ 0 & I_6 \end{pmatrix}, \tag{14}$$

$$Q(\delta) = \begin{pmatrix} \frac{\delta^3}{3} I_6 & \frac{\delta^2}{2} I_6 \\ \frac{\delta^2}{2} I_6 & \delta I_6 \end{pmatrix}. \tag{15}$$

The measurement function for updates by the edge-based tracker is the projection of the state onto the six pose parameters because the motion vector μ is already calculated in the camera frame around the prior camera pose. Thus, the measurement Jacobian matrix is the identity and the update follows the standard EKF formulas with the innovation described by measurement vector μ and the measurement covariance matrix C from equation (8).

Similarly, measurements of the rotational velocity of the gyroscope sensor are made in the camera frame. Therefore the measurement function is again only a projection of the state vector onto the rotational velocities. The covariance matrix of the measurement noise was determined experimentally.

Both magnetic field and linear acceleration are modeled as static 3D vectors representing the direction of the local magnetic field $\boldsymbol{m}_w$ and gravity $\boldsymbol{g}_w$ in the world. The measurement functions are

$$\boldsymbol{m}_c = R_t^- \boldsymbol{m}_w \tag{16}$$

$$\boldsymbol{g}_c = R_t^- \boldsymbol{g}_w \tag{17}$$

where R_t^- is the prior estimate of camera rotation at time t. The measurement Jacobian is evaluated using the exponential map parametrisation of R_t^-. The covariance matrices of the respective measurement noise distributions were again determined experimentally. Modelling linear acceleration as a static gravity vector is a coarse simplification, however it works well using an increased measurement noise to hide the model errors.

Sensor data also needs to be transformed into the camera frame. A rotation R_{cw} of the camera corresponds to a rotation $R_{sw'}$ of the sensor in its own world frame w' by

$$R_{cw} = R_{cs} R_{sw'} R_{w'w}, \tag{18}$$

where R_{cs} is the rotation from the local sensor to the camera coordinate system and $R_{w'w}$ the rotation from the world w to the sensor world w' coordinate system. Both R_{cs} and $R_{w'w}$ were calibrated using the method described in [3]. Sensor data itself is transformed using R_{cs} from the sensors coordinate system s into the camera coordinate system c. The reference vectors $\boldsymbol{m}_{w'}$ and $\boldsymbol{g}_{w'}$ which were estimated offline from sensor data are converted into the world coordinate system w by the rotation $R_{ww'} = R_{w'w}^{-1}$.

The sensor pack delivers gyroscope, acceleration and magnetic field measurements at 100Hz. These are queued with video frames

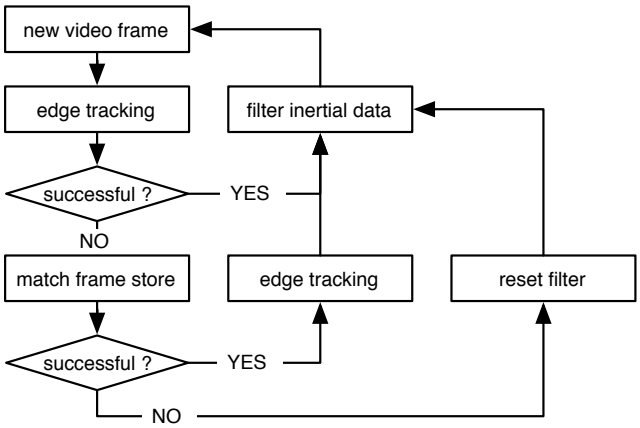

Figure 3: Control flow between components. After matching to the frame store the pose estimate is refined with another iteration of edge-tracking.

to be incorporated into the filter state in the correct order given by their respective time stamps. An offset between the time stamps for video frames and for sensor pack measurements was calibrated offline by comparing the orientation estimates from just optical tracking with those derived from sensor pack measurements alone. The measurements are added collectively as a single 9 dimensional measurement with the measurement Jacobian being a combination of the individual Jacobians.

4 Recovery

The tracking system described above exhibits acceptable performance as long as the view of the environment is not obstructed by unmodelled objects. Transient occluders such as cars and buses driving past or pointing the camera towards unknown areas (the sky) break the assumptions of the edge-based tracker. Inertial and magnetic sensors only provide orientation measurements and do not allow to estimate translational movements during occlusions. Additionally, as the system does not perform global data association, it can easily lock onto edges of transient occluders and fail quickly. Therefore, an additional recovery mechanism is required to improve the overall robustness of the system.

The principle of operation behind the recovery component is twofold. After pose optimisation, a statistical test detects when the edge-based tracking system fails. In the case of failure, the recovery component tries to match the current video frame to a frame store of older frames with known camera pose recorded online during system operation. If the matching succeeds, the new pose estimate is based on the motion between the stored frame and the current video frame. Otherwise, the filter's velocities are reset to zero and the camera pose estimate is not fused into the filter state to avoid introducing a wrong pose and velocity estimate. The same procedure is repeated on the next frame (see Figure 3). The frame store itself is populated online with frames and corresponding poses selected from frames where the corresponding pose was well-estimated.

4.1 Quality-of-tracking test

The NCC based edge detection process culls edge associations with a low score to prevent outlier measurements from entering the optimisation. The percentage of such failures was measured for a set of sequences where tracking was working correctly as determined by visual inspection, yielding a failure propability p_+. A similar measurement was performed for a second set of sequences where

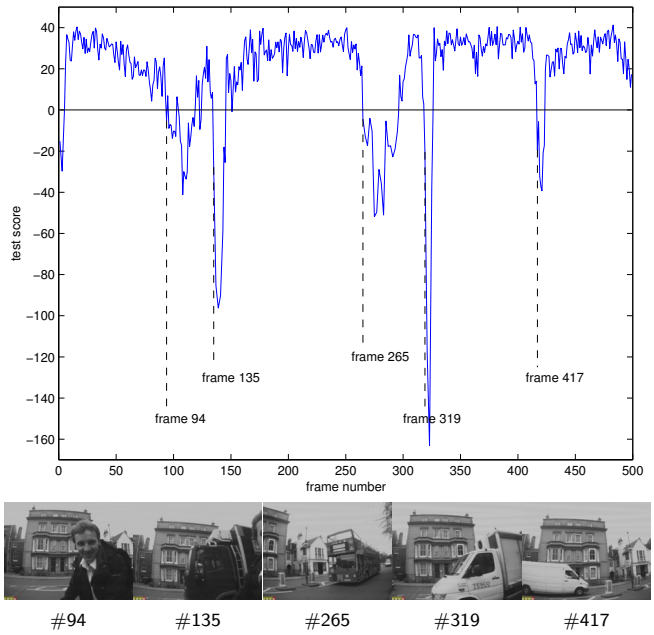

Figure 4: The plot shows the development of the test score computed in equation (19) in a sequence of 500 frames. The images show frames where the score dropped below 0. Frame #94 a colleague is blocking the view, #135 a truck drives up behind him, #265 a bus comes along, #319 another truck and #417 a van in the opposite lane.

tracking was failing, yielding a failure probability $p_- > p_+$. Then for a given video frame with F failures among N samples, we compute the log ratio r of the evidence for p_+ and p_- as

$$r = \ln \frac{P(N,F|p_+)}{P(N,F|p_-)} = \ln \frac{\binom{N}{F} p_+^F (1-p_+)^{N-F}}{\binom{N}{F} p_-^F (1-p_-)^{N-F}} = F \ln \left(\frac{p_+}{p_-}\right) + (N-F) \ln \left(\frac{1-p_+}{1-p_-}\right). \quad (19)$$

If r becomes negative, the system assumes that the edge-based tracker did not converge to a correctly matched pose. Figure 4 shows the test scores for a sample sequence with video frames at crucial points.

4.2 Point matching against the frame store

The frame store component retains a set of video frames recorded during operation and corresponding camera poses as estimated by the filter. For each such frame, a set of corner-like features are extracted using the FAST corner detector [23]. A feature vector f corresponding to the 16 pixel intensities located on the ring of pixel locations used in the FAST detector is extracted and stored for each point. The frame store accepts only frames with tracking quality r above a threshold and with a minimal separation in time, to be able to cover a certain range of the past, and avoid cluttering the store with consecutive frames containing the same information. In addition, it removes old frames to keep a fixed number of frames. In our implementation stored frames have to be separated by at least 1 second and a maximum of 10 frames is stored. The feature detector yields between 200 and 400 features per frame.

To match a query frame against the stored frames, a set of features F_q is extracted from it as well. Then set-to-set matching be-

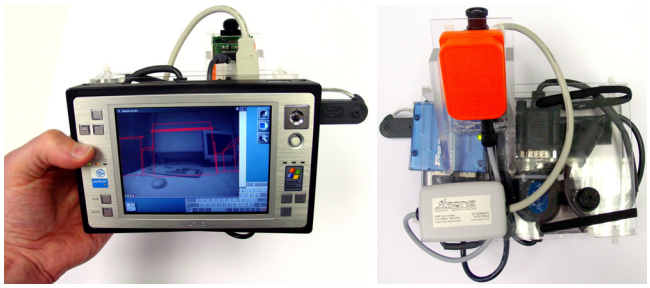

Figure 5: Images of the handheld AR device. A camera (circuit board and black lense) and inertial sensor pack (orange cube) are mounted to the back of the device.

tween the feature set of the query frame F_q and the set of each stored frame F_s is performed. The features are compared using sum-of-squared-distances (SSD) of the feature vectors and the best match for each feature is stored. The method described in [23] is employed to avoid the full $O(N^2)$ cost of matching every feature in F_q against every other feature in F_s. Features in F_s are sorted by the mean value $\overline{f}$ of the feature vectors. When matching a feature $f_q \in F_q$, the feature $f_s \in F_s$ with the closest mean is found using binary search. Starting from this feature a linear search in both directions is performed. The search can be terminated as soon as the bound for the SSD,

$$SSD(f_q, f_s) \geq n(\overline{f_q} - \overline{f_s})^2 \tag{20}$$

where n is the number of dimensions in the feature vectors, is exceeding the best match. Only match pairs where the corresponding features provide the best match for the other one are kept (so called 'married matches'). This yields a high fraction of inliers, simplifying pose estimation.

The resulting set of point correspondences is used to compute the camera motion between the stored frame and the query frame. We are interested only in a pose estimation that allows the edge tracker to converge again. Therefore, we compute a pose that assumes the same position as the stored frame but optimises the orientation to fulfil the 2D feature locations in the query frame. Edges reprojected under the resulting pose are then close enough to the real edges and edge-tracking can lock on again.

To compute this pose we estimate a 3D rotation between the stored frame and the query frame. Using RANSAC on subsets of 4 pairs of 2D correspondences, we estimate a 2D homography between the point pairs which is constrained to a 3D rotation of the camera frame. After a fixed set of iterations, the inlier set of the hypothesis having the largest support is used to estimate the final 3D rotation.

The stored frame and optimised rotation yielding the lowest average re-projection error are selected as the best match and the new pose is calculated by multiplying the frame pose with the 3D rotation. This pose is then used as the prior for another iteration of edge-based tracking to allow the tracker to lock on again. The pose estimated in this way may differ from orientation measurements provided by the sensor pack. However, the subsequent edge-based tracking converges back to the true pose which agrees with the measurements.

5 Demonstration Application

We have implemented a simple location based game using a handheld AR platform to demonstrate the performance of the tracking system. The platform consists of a VAIO U-71 tablet PC with 1.1 GHz Pentium III M CPU. A USB 2.0 camera with a 3.6mm lense

Figure 6: From top left in clockwise order: Select your sweetheart, on the way to pick up the ladder, close to the ladder now, successful delivery through the window.

and an Xsens MTx sensor are mounted rigidly to the back of the device. Figure 5 shows details of the system.

A system of this size lies between the tablet-based systems [16] and stand-alone PDA-based systems [30]. While computing power is similar to the tablet-based platforms, the smaller size and reduced weight make it a possible substitute for a PDA-based system.

The camera captures video frames with a resolution of 640×480 at 30 Hz. However, the tracking components use only a quarter frame of resolution 320×240 to operate at interactive frame rates. The system is able to perform at 15–17 Hz using these parameters.

The tracking system itself does not handle initialisation and thus the user has to start from a well known point in the environment. The startup is simplified because the acceleration and magnetic sensors provide absolute orientation measurements and therefore the system has a good initial estimate of its orientation. Furthermore, it uses the quality-of-tracking test (see section 4.1) to determine automatically when to start tracking. The resulting operation is very simple: the user points the handheld device at any part of the environment represented in the model while standing at the starting point and the system picks up tracking automatically. Alternatively, the orientation measurements can be discarded and the user has to align the view from the initial position and orientation with the live image to allow the system to start tracking.

Overlaid computer graphics are rendered with radial distortion compensation using a vertex program to adjust vertex locations within the OpenGL pipeline. Therefore, the accuracy depends on the tessellation of the rendered objects as only vertices are modified and not individual fragment locations. The standard texture based approach was not applicable, because the overhead of one texture transfer would reduce the system's performance unacceptably.

The game itself tries to capture the Alpine tradition of 'Fensterln', an activity where lovers deliver notes directly through a sweetheart's window. The goal of the game is to locate the right window in the environment and reach it via a ladder. First a ladder has to be found and picked up by walking up to it. Then walking to the ground below the window, the ladder is placed against the wall and the note climbs up the ladder and vanishes into the room beyond the window. However, if one fails to find the ladder and window in time, one's love is lost forever. Figure 6 shows some impression of the game.

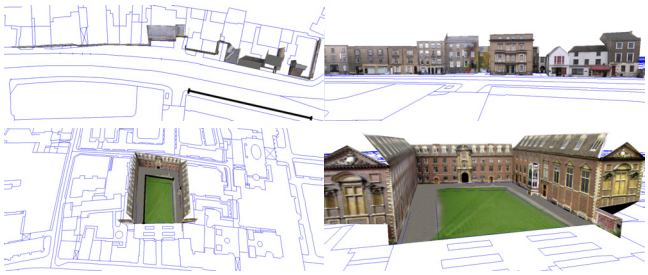

Figure 7: Overhead and frontal views of the models for Trumpington Street and St. Catherine's College main court. The blue lines are vector map features registered with the models. Black lines in the overhead views denote trajectories for evaluation against map data.

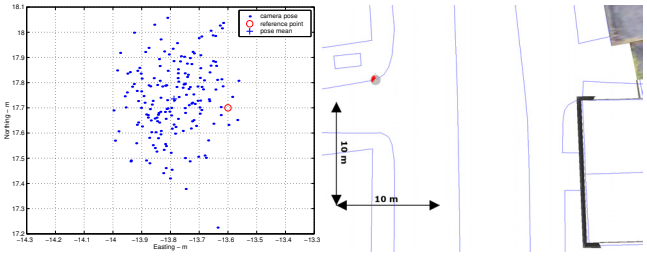

Figure 8: (Left) Distribution of camera coordinates (x,z) for a stationary camera. The cross denotes the mean, the circle denotes the reference point from the map. (Right) Plot of the camera poses in the 3D model. The yellow dot denotes the map reference point, the red cloud the camera poses and the gray disk describes a circle of 0.5m radius around the reference point.

6 Results

Two sites were modelled to provide test environments for evaluations of the system's performance. The first location is on Trumpington street in the City of Cambridge. The model contains a set of building facades and was created by Duncan Robertson using the Photobuilder software [21] and kindly made available for experiments. The second site is the main court of St. Catherine's College, modelled using a commercial software package. Both models were registered with vector map data obtained from the Ordnance Survey agency to provide ground truth for accuracy evaluations. Figure 7 shows overhead and frontal views of the two models with registered map data shown as blue lines.

The models capture the overall shape of buildings as large planar surfaces with highly detailed textures. Small scale facade elements and building structures are not modeled. Such differences contribute to errors in the absolute pose estimate of the camera. The registration between 3D models and map points were computed as rigid transformation plus uniform scale to avoid distortions of the structures. Because the 3D models created from photographs do not exactly fit the corresponding map points, the registration still contains errors of up to 0.5m between points in the models and the map data.

6.1 Accuracy

To evaluate accuracy of the localisation, we compair our results with ground truth data delineated with respect to the Ordnance Survey map. For a 3D point (x,y,z), the coordinate x corresponds to the easting of the map, y to elevation above the ground plane and z to the northing of the map (easting and northing describe the coordinate axes in a map with a unit of 1m). Figure 8 shows camera coordinates (x,z) for a sequence where the camera is held stationary close to the ground above a known map point and only the edge tracker was used to estimate pose. The standard deviations in the three position coordinates are $(\sigma_x,\sigma_y,\sigma_z) = (0.0979m, 0.1577m, 0.1463m)$. The difference of the mean $(\hat{x},\hat{y},\hat{z}) = (-13.7876m, 0.4476m, -17.7354m)$ to the known map point at $(-13.6m, 0m, 17.7m)$ can be attributed to inaccuracies in the 3D model such as missing structures on the facade and errors in the registration between the 3D model and the map data.

To evaluate the accuracy under motion, we walked along the trajectories denoted in Figure 7 and recorded the estimated camera pose. The trajectory in the Trumpington street model follows the curb edge of the side walk. Figure 9 shows plots of the camera coordinates in (x,z) and the distance of the pose estimation from the sidewalk line over time. The systematic error between frame #200 and #350 results from protrusions on the central building's facade which were not modelled. Hence, the building appears to be closer than it is and the camera trajectory deviates from the line.

Figure 10 shows the plot of a 2 min sequence of walking through the main court of St. Catherine's College. After walking along the grass centre, the camera rotates in various directions to evaluate the performance under close proximity to modelled structures. In these situations tracking fails at several points due to larger discrepancies between the flat textured model and the actual building. However, automatic recovery is possible through the frame store. These recoveries are evident as sudden large jumps in position, as the reference position from an earlier frame is used. The pose estimate quickly converges back to the true position. Closer to the end of the sequence, the recovery cannot obtain a good position anymore and the tracking fails.

6.2 Robustness

The system successfully handles transient disturbances that break simple tracking solutions. Complete occlusion of the view by passing vehicles or persons can be tolerated because the recovery mechanism obtains a pose as soon as the view resembles a stored image again. Figure 11 shows typical disturbances and the subsequent recovery of the system for the same sequence as in figure 4.

The frames selected by the test score as tracking failures appear to be good enough for tracking to work. The test overestimates failures, because it detects that the image does not correspond to what the selected edge profiles predict. This is always true for failure conditions of the edge-based tracker, but may also hold for images which were tracked successfully. However, overestimation of failures is advantageous as recovery from outlier motion and velocity states in the filter is more difficult than from losing a motion update and setting velocity to zero.

The recovery mechanism cannot replace a real initialisation algorithm and has limitations in its current form. The selection of reference frames should also take spatial distribution of the associated camera poses into account to gather a more representative set of frames, instead of the current temporal distribution. The feature matching is not invariant to camera motions around the optical axis, reducing the number of possible fits under such motions. Finally, the estimation of the camera motion between reference frames and query frames should take translation into account as well by computing the epipolar geometry between the two frames.

6.3 Performance

The system currently operates at about 15-17 frames per second with small drops in frame rate to about 13 FPS upon access to the frame store. Table 1 gives an overview of the processing times of individual steps. Retrieving the frame and depth buffers from the

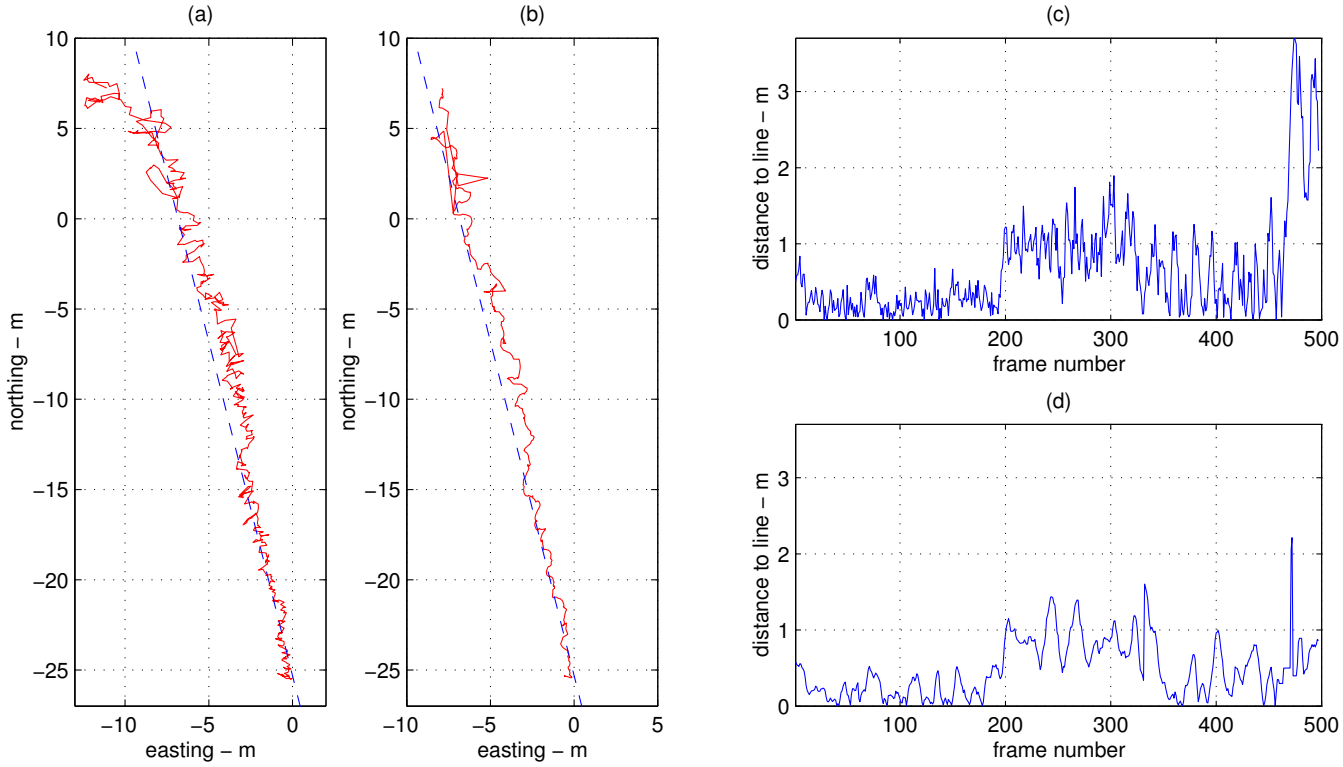

Figure 9: (a) Plot of easting and northing of the camera pose as estimated by the edge-based tracker without filtering. The blue stippled line denotes the reference edge of the sidewalk that the pose should lie on. (b) Plot of the Kalman filter estimates incorporating inertial sensors as well. (c) Distance to the reference edge for plot (a). (d) Distance to the reference edge for plot (b).

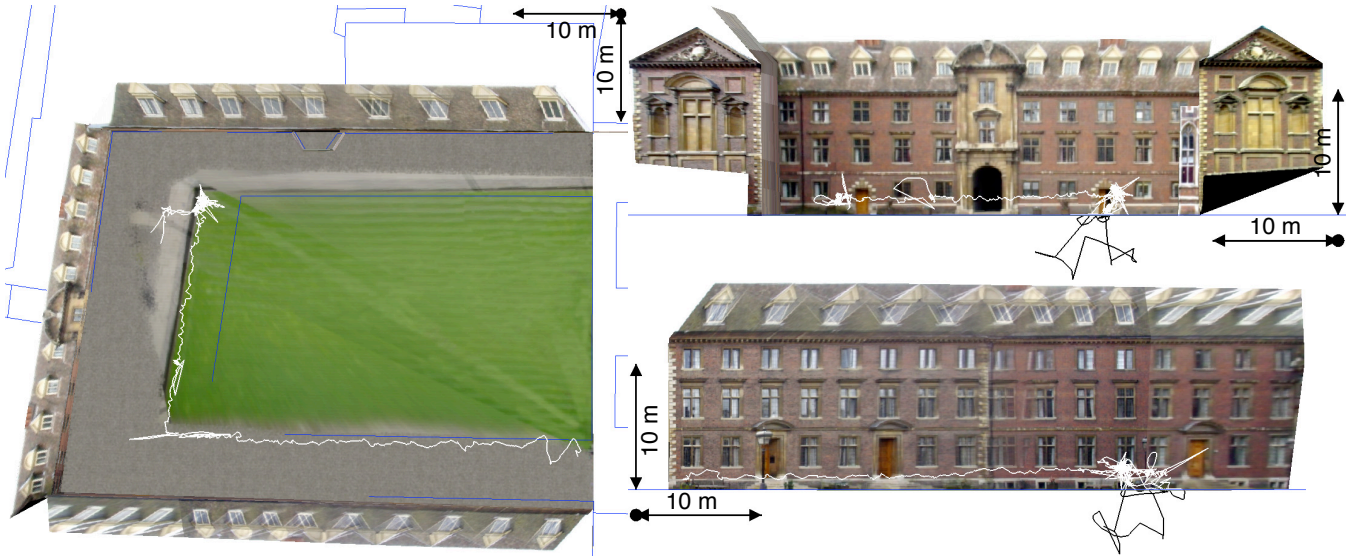

Figure 10: Overhead, front and side plots of camera centre location from a sequence of walking through St. Catherine's College main court. The walk starts in the lower right corner, along the grass centre and finishes with a set of pans in the upper left corner. Tracking fails during the last rotations leading to wrong pose estimates. Individual large deviations are due to incorrect position estimates from a reference frame, but converge quickly to the right location again.

Figure 11: Examples of recovery from occlusions. The first column shows the first image where the test score drops below zero. The second column shows the first image that matched to a reference image (third column) and from which tracking succeeded again. Grey dots denote point features and white lines inlier correspondences under a rotation. The forth column shows the correctly overlaid wireframe again. Rows are: failure at frame #94 and recovery at frame #121, failure #265 and recovery #292, failure #319 and recovery #328, failure #417 and recovery #423.

OpenGL system incurs the largest overhead. Newer graphics hardware improves the system's performance substantially.

While the tracking operation alone can run at 25 Hz, other processing steps such as converting and downsampling the input frame and drawing the video background and user interface incur sufficient overhead to lower the system frame rate to 17Hz.

individual step	time in ms
render 3D model	2.25
read frame buffer	15.89
read depth buffer	8.05
edgel extraction	4.90
edge search	2.80
total cost (including two iterations of edge search)	40.16
frame grab and subsampling	1.52
drawing background and user interface	15.29
matching against frame store (10 frames)	22.40

Table 1: Average processing times for individual steps of the system.

6.4 Dynamic behaviour

The video accompanying the paper shows the system in operation. Due to the combinations of inertial sensors and the recovery mechanism it is quite robust and avoids common failure modes such as fast motion, total occlusion, or misalignments of the edge-based tracker. While its absolute localisation performance as described by the variance and errors in absolute pose is not perfect, the quality of overlays is not affected, because the errors in image space are minimised, which is most important for convincing augmented reality applications. Visible errors in the graphics overlays are mostly due to the vertex-based radial distortion compensation.

When the distance between tracked surfaces and the camera is too small the assumption of planarity underlying the 3D model is no longer valid. This reduces good edge matches (because facade features change their appearance strongly under different viewing angles) and introduces errors in the depth localisation of edgels in the back projection step. Both types of errors contribute to the failure of the system in such situations. Nevertheless, the range of distances between camera and model that can be tracked successfully is encouraging.

7 Conclusions

This paper presents a robust visual tracking system for handheld augmented reality in urban environments. An established edge-tracking technique is extended to textured 3D models and appearance based-line detection. A complementary recovery component based on point-based image matching into a database of reference frames allows for recovery from failures of the edge tracker. The resulting system substantially improves both accuracy and robustness over former tracking solutions for similar environments. The accuracy evaluation using map based ground truth data gives objective estimation of the system's performance.

The basic idea of the presented approach is to exchange the step of creating the primitives (here lines) used in detection with the step of calculating the visibility and additional appearance of the primitives. Creating the line model first and computing visibility for lines represents the traditional edge-based approach. Our approach first computes a rather generic view of the world from which the lines and edgels are extracted on the fly. While the former approach reduces stored information as early as possible, the advent of modern graphics hardware makes it possible to retain more information until the rendering stage. This additional information creates a more faithfull representation of visible lines which helps to select edgels for tracking and improves detection.

The tracking system could be applied to other display types as well. Head mounted displays using video see-through overlays can directly use the same techniques, while optical see-through displays would require additional calibration of the HMD's virtual camera with respect to the video camera.

Initialisation is an important topic for future work. A handheld GPS unit can provide a basic, but inaccurate estimate of position that can be used to either index into a database of reference frames established offline, or to limit the search range for an online raster search of possible locations. Such an extension is the next step toward a fully functional system for accurate urban localisation.

The platform itself will be integrated with a stationary command & control information visualisation system. This combination should support collaboration between ground personnel equipped with the handheld unit and staff working in the control centre. The resulting asymmetry in the information presentation capabilities of the two environments is a topic of further research.

Acknowledgements

The authors thank Dr. Duncan Robertson and Prof. Roberto Cipolla for making the model of Trumpington Street available for experiments. This work was supported by a grant from the Boeing Corporation.

References

[1] Ronald Azuma, Bruce Hoff, Howard Neely, and Ron Sarfaty. A motion-stabilized outdoor augmented reality system. In *Proc. IEEE VR*, pages 252–259, 1999.

[2] Y. Baillot, D. Brown, and Simon Julier. Authoring of physical models using mobile computers. In *Proc. ISWC 2001* [10], pages 39–46.

[3] Yohan Baillot, Simon J. Julier, Dennis Brown, and Mark A. Livingston. A tracker alignment framework for augmented reality. In

Proc. ISMAR 2003, pages 142–150, Tokyo, Japan, October 7–10 2003. IEEE.

[4] Reinhold Behringer. Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors. In *Proc. IEEE VR'99*, Houston, Texa, USA, March 13–17 1999.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[6] Volker Coors, Tassilo Huch, and Ursula Kretschmer. Matching buildings: Pose estimation in an urban environment. In *Proc. ISAR 2000*, pages 89–92, Munich, Germany, October 5–6 2000. IEEE and ACM.

[7] Tom W. Drummond and Roberto Cipolla. Visual tracking and control using lie algebras. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition 1999*, Ft. Collins, CO, USA, June 23–25 1999. IEEE.

[8] Steven Feiner, Blair MacIntyre, Tobias Höllerer, and Anthony Webster. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban enviroment. In *Proc. ISWC'97*, pages 74–81, Cambridge, MA, USA, October 13–14 1997.

[9] Christian Fruh and Avideh Zakhor. Constructing 3d city models by merging aerial and ground views. *IEEE Comp. Graph. Appl.*, 23(6):52–61, Nov/Dec 2003.

[10] Tobias Höllerer, Steven Feiner, Tachio Terauchi, Gus Rashid, and Drexel Hallaway. Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computer & Graphics*, 23(6):779–785, 1999.

[11] Jinhui Hu, Suya You, and Ulrich Neumann. Approaches to large-scale urban modeling. *IEEE Comp. Graph. Appl.*, 23(6):62–69, Nov/Dec 2003.

[12] Bolan Jiang, Ulrich Neumann, and Suya You. A robust tracking system for outdoor augmented reality. In *Proc. VR 2004*, pages 3–10, Chicago, USA, March 27–31 2004. IEEE.

[13] Björn Johansson and Roberto Cipolla. A system for automatic pose-estimation from a single image in a city scene. In *IASTED Int. Conf. Signal Processing, Pattern Recognition and Applications*, Crete, Greece, June 2002.

[14] Hirokazu Kato and Mark Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferenencing system. In *Proc. IWAR'99*, pages 85–94, San Francisco, CA, USA, October 21–22 1999. IEEE CS.

[15] Georg Klein and Tom W. Drummond. Robust visual tracking for non-instrumented augmented reality. In *Proc. ISMAR 2003*, pages 113–122, Tokyo, Japan, October 7–10 2003. IEEE.

[16] Georg Klein and Tom W. Drummond. Sensor fusion and occlusion refinement for tablet-based ar. In *Proc. ISMAR 2004*, pages 38–47, Arlington, VA, USA, November 2–5 2004. IEEE.

[17] Jong Weon Lee, Suya You, and Ulrich Neumann. Tracking with omni-directional vision for outdoor ar systems. In *Proc. ISMAR 2002*, pages 47–56, Darmstadt, Germany, September 30 – October 1 2002. ACM and IEEE.

[18] Vincent Lepetit, Luca Vacchetti, Daniel Thalmann, and Pascal Fua. Fully automated and stable registration for augmented reality applications. In *Proc. ISMAR 2003*, pages 93–102, Tokyo, Japan, October 7–10 2003. IEEE.

[19] Wayne Piekarski and Bruce H. Thomas. Tinmith-metro: New outdoor techniques for creating city models with an augmented reality wearable computer. In *Proc. ISWC 2001*, pages 31–38, Zurich, Switzerland, 8–9 October 2001. IEEE.

[20] Miguel Ribo, Peter Lang, Harald Ganster, Markus Brandner, Christoph Stock, and Axel Pinz. Hybrid tracking for outdoor augmented reality applications. *IEEE Comp. Graph. Appl.*, 22(6):54–63, 2002.

[21] Duncan P. Robertson and Roberto Cipolla. Building architectural models from many views using map constraints. In *Proc. 7th European Conference on Computer Vision, LNCS 2351*, volume 2, pages 155–169, Copenhagen, Denmark, May 2002. Springer Verlag.

[22] Duncan P. Robertson and Roberto Cipolla. An image-based system for urban navigation. In *Proc. BMVC 2004*, London, UK, September 7–9 2004.

[23] Edward Rosten and Tom W. Drummond. Fusing points and lines for high performance tracking. In *Proc. ICCV 2005*, pages 1508–1511, Beijing, China, October 15–21 2005.

[24] Kiyohide Satoh, Kentaro Hara, Mahoro Anabuki, Hiroyuki Yamamoto, and Hideyuki Tamura. TOWNWEAR: An outdoor wearable MR system with high-precision registration. In *Proc. IEEE Virtual Reality 2001*, pages 210–211, Yokohama, Japan, March 13–17 2001.

[25] Gilles Simon, Andrew W. Fitzgibbon, and Andrew Zisserman. Markerless tracking using planar structures in the scene. In *Proc. ISAR 2000*, pages 120–128, Munich, Germany, October 5–6 2000. IEEE and ACM.

[26] Iryna Skrypnyk and David G. Lowe. Scene modeling, recognition and tracking with invariant image features. In *Proc. ISMAR 2004*, pages 110–119, Arlington, VA, USA, November 2–5 2004. IEEE.

[27] Didier Stricker. Tracking with reference images: A real-time and markerless tracking solution for out-door augmented reality applications. In *Proc. VAST 2001*, Glyfada, Athens, Greece, November 28–30 2001. Eurographics.

[28] Bruce H. Thomas, V. Demczuk, Wayne Piekarski, David Hepworth, and Bernard Gunther. A wearable computer system with augmented reality to support terrestrial navigation. In *Proc. ISWC'98*, pages 168–171, Pittsburgh, PA, USA, October 19–20 1998. IEEE and ACM.

[29] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *Proc. ISMAR 2004*, Arlington, VA, USA, November 2–5 2004. IEEE.

[30] Daniel Wagner and Dieter Schmalstieg. First steps towards handheld augmented reality. In *Proc. ISWC 2003*, White Plains, NY, USA, October 21–23 2003.

[31] Greg Welch and Gary Bishop. Scaat: incremental tracking with incomplete information. In *Proc. SIGGRAPH '97*, pages 333–344, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[32] Harald Wuest, Florent Vial, and Didier Stricker. Adaptive line tracking with multiple hypotheses for augmented reality. In *Proc. ISMAR 2005*, pages 62–69, Vienna, Austria, October 5–8 2005. IEEE and ACM, IEEE CS.

[33] Suya You, Ulrich Neumann, and Ronald Azuma. Orientation tracking for outdoor augmented reality registration. *IEEE Comp. Graph. Appl.*, 19(6), November 1999.