Online Submission ID: 115

Implicit 3D Modeling and Tracking for Anywhere Augmentation

Abstract

This paper presents an online 3D modeling and tracking methodology that uses aerial photographs for mobile augmented reality. Instead of relying on models which are created in advance, the system generates a 3D model for a real building on the fly by combining frontal and aerial views with the help of an optical sensor, an inertial sensor, a GPS unit and a few mouse clicks. A user's initial pose is estimated using an aerial photograph, which is retrieved from a database according to the user's GPS coordinates, and an inertial sensor which measures pitch. To track the user's position and orientation in real-time, feature-based tracking is carried out based on salient points on the edges and the sides of a building the user is keeping in view. We implemented camera pose estimators using both a least squares and an unscented Kalman filter (UKF) approach. The UKF approach results in more stable and reliable vision-based tracking. We evaluate the speed and accuracy of both approaches, and we demonstrate the usefulness of our computations as important building blocks for an Anywhere Augmentation scenario.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.4.8 [Image Processing and Computer Vision]: Scene Analysis

Keywords: Outdoor augmented reality, online modeling, feature-based tracking, UKF, camera pose estimation

1 Introduction

Augmented Reality (AR), which makes the physical world a part of the user interface experience, has the potential to play a significant role in enhancing the mobile and wearable computing paradigm. "Anywhere Augmentation" provides a conceptual extension of Mobile Augmented Reality (MAR) [Höllerer et al. 2007]. Its aim is to enable the linking of location-specific computing services with the physical world, making them readily and directly available in any situation and location. A user should be enabled to augment and interact with virtual objects in unprepared environments without the need for prolonged initial setup. Furthermore, the approach relies only on globally available data sets, such as aerial photographs of the user's surroundings. To realize "Anywhere Augmentation", it is essential to estimate an accurate pose of a user in real-time, even in unprepared environments.

Up to now, a wide variety of tracking technologies, employing various sensors, have been investigated for AR systems [Azuma et al. 2001]. Model-based vision tracking technologies have demonstrated the highest quality results among standard vision techniques currently applied in most AR applications [Reitmayr and Drummond 2006]. Model-based tracking relies on appropriate features, e.g., points or edges in textured 3D models. However, creating 3D models exhibiting such detailed features for large environments becomes a daunting task. Moreover, models to be tracked have to be prepared in advance before tracking can begin. In other words, a user cannot use AR in unprepared environments.

The proposed approach does not depend on an a-priori model, but instead employs an online 3D modeling technique for simple building geometry in unprepared environments. In this way it shares general goals with Davison's Mono-SLAM work [2003], but in order to be more generally feasible in outdoor environments, it employs simple human input and additional generally available information. A video image and an aerial photograph are used in combination based on the estimates from a GPS and an inertial sensor to create a virtual 3D building model on the fly. Only simple user interactions, i.e. mouse clicks on both the video image and the aerial photograph are needed to associate the frontal and aerial views. Then, salient features on the edges and sides of a building are extracted for real-time tracking. For stable and reliable tracking over time, the UKF is employed.

We present the following contributions in this paper: First, we introduce a simple initialization procedure that models building corners with information from the aerial photograph and the user's view. This is assisted by semi-automatic edge and corner detection to bring down the necessary interaction burden to three mouse clicks. Second, we present camera pose tracking for a moving user, using a set of salient feature points from the model of a building corner, the adjacent edges and building walls. We allow the user to place annotations of building features (e.g. windows, doorways or historic facade details) and stabilize these accurately in the user's view. Finally, we compare results for this vision-based tracking approach using two different estimators: least squares and UKF. All these steps are important components for being able to model and annotate the 3D world on the fly in unprepared environments, in the spirit of Anywhere Augmentation.

The rest of this paper is structured as follows: After a discussion of related work in Section 2, the steps for online 3D model generation are presented in Section 3. In Section 4, we introduce the camera tracking module based on the generated 3D model and UKF. Section 5 demonstrates experimental results regarding speed and performance of the system. We present our conclusions and ideas for future work in Section 6.

2 Related Work

One of the main problems with current approaches to mobile AR is that in order to obtain reliable and accurate registration between the physical world and the augmentations one either needs a model of the environment, or the environment needs to be instrumented, at least passively with registration markers. Both of these preconditions severely constrain the applicability of AR. Early mobile AR work mostly relied on open loop tracking and a pre-registered environment model. Early examples of this are the Touring machine [Feiner et al. 1997], the Battlefield Augmented Reality System [Baillot et al. 2001], and the Tinmith system [Piekarski and Thomas 2001]. These systems relied on GPS and inertial and/or magnetic orientation sensors. Vision-based tracking systems have become increasingly feasible in the past ten years, even though vision-only approaches are not sufficiently general and robust yet to work in unprepared and unmodeled environments – the focus of this paper.

For real-time pose estimation, several kinds of image features, such as points, lines, contours or a combination of these different geometric primitives, have been researched as follows. Behringer [1999] proposed horizon silhouette matching for improved the orientation tracking. Coors et al. [1999] proposed a localization system where a video image is compared with an underlying GIS based on extracted edges. Simon et al. [2000] described a markerless camera tracking system for an outdoor AR system that capitalizes on the occurrence of planar structures in the environment. Ribo et al. [2002] employed feature points for their hybrid outdoor tracking system. Vacchetti et al. [2004] presented a real-time 3D tracking approach that combines edges and point-of-interest features. They employed a multiple hypotheses technique to enable to use the edges information even when

the features to track are much weaker than the misleading features in the background. Similarly, Rosten & Drummond [2005] proposed a combination of edge and point tracking to increase robustness under fast camera motions. Reitmayr et al. proposed a model-based hybrid tracking system for outdoor AR, enabling accurate, real-time overlays for a handheld device, using a textured 3D model [2006]. The system tracks features under fast motion, combining their vision-based predictions with gyroscope measurements in an extended Kalman filter framework. Their computer vision approach goes back to methods developed by Drummond & Cipolla [1999] and Klein and Drummond [2003].

All of these previous methods calculate camera pose based on the correspondences between 2D image features and their 3D coordinates. In other words, they assume an a-priori 3D model before tracking. A notable exception is work in the area of simultaneous localization and mapping (SLAM). Davison and colleagues brought the SLAM approach from the domain of mobile robotics to hand-held or head-worn camera applications [Davison 2003][Davison et al. 2003]. Our work is combining aspects of Mono-SLAM and model-based AR tracking approaches. The goal is to model concrete semantic entities (buildings) directly, without first accumulating abstract 3D feature clouds. This is only feasible if the human operator provides some semantic knowledge about the environment (in our case by picking building corners with adjacent planar wall structures and by establishing correspondences with the ubiquitous "2D world model" provided by aerial photographs).

Maps and aerial photographs have also been employed in many mobile systems for passive localization purposes. ARVino used aerial photographs in a virtual reality view of GIS data to assist a user in mentally mapping abstract information onto the physical environment the data annotates [King et al. 2005]. Wither et al. also presented a mobile AR system for outdoor annotation of the real world by employing aerial photographs in addition to the wearable system's data sources, such as position, orientation, camera and user input [2006]. They enabled a user to annotate 3D features with a few interactions by aligning features in the first person viewpoint and in the aerial view. We extend the functionality of the above systems by using aerial photographs in a 3D model generation for tracking as well as localization of a user.

3 Online 3D Modeling

We are working towards a seamless AR experience in unprepared environments for which no 3D model exists yet. We propose methodology for online 3D model generation and tracking of the observer's pose using those emerging models. To this end, we present a method for modeling building corners using a GPS, an inertial sensor, a video-see-through head-worn display, and access to aerial photographs of the user's environment. Additionally, a tracking method based on salient features in the user's view of the model is presented, using UKF as the predictor framework. The whole procedure is divided into two parts; online 3D modeling and camera pose tracking. The flow diagram for the whole procedure is depicted in Figure 1.

In the online 3D modeling module, an initial pose is estimated using GPS and an inertial sensor. According to the user's position, an appropriate aerial photograph section is retrieved from an online database. Our current prototype's database only covers our campus and its surroundings, but databases of national and global scale exist and could be adopted [Google 2007; Yahoo 2007]. The accuracy of the GPS and orientation sensors is limited, but we can subsequently increase accuracy using the aerial photograph and human calibration as described by [Wither et al. 2006].

To model a corner part of a building, a user is looking at the building corner through a camera. The camera orientation at that moment is captured by the inertial sensor. The corner of the building is then calculated as the intersection point of three lines passing through the corner in the user's camera image. The corner detection step is also

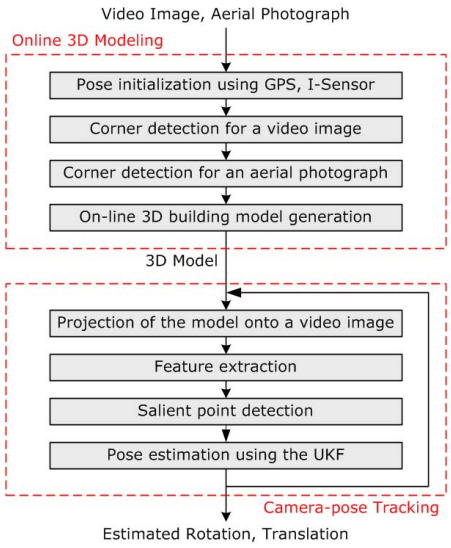


Figure 1: *An overall flow diagram*

applied to an aerial photograph to calculate the distance between the corner of the building and the user's known position. The height of the building can thus be estimated from the distance and the orientation. Using the above approach, it is possible to generate a partial 3D building model on the fly. For the purpose of this paper, we assume box-shaped building corners.

As a second step, in the camera pose tracking module, the generated 3D model is projected onto a video image from a current pose estimate. Then, features on walls are extracted and tracked. At every visible control point, one-dimensional edge detection in the direction of the model edge normal is performed in the video image to find the corresponding salient points. The vector between the control point and the closest intensity edge, i.e. the salient point, detected within a cutoff distance of the control point is defined as an image-space correction vector. The camera pose is updated by a vector of motion parameters to minimize these image errors using a UKF framework for stable and reliable tracking.

3.1 Yaw estimation with GPS and an aerial photograph

Our approach takes advantage of a local planar coordinate system for the spatial extent retrieved from the aerial photograph database. We calculate the local coordinates from the longitude and latitude provided by GPS through Universal Transverse Mercator Projection (UTM) [Hofmann-Wellenhof et al. 1997], employing the most suitable reference ellipsoid given the user's geographic location.

After the estimation of a user's position in the local coordinate system, we need to establish the user's head orientation with regard to the building's orientation in the local coordinate system. The Intersense InertiaCube2 orientation tracker we use has a built-in magnetometer that establishes magnetic north as an absolute direction, from which geographic north can be determined based on the current longitude and latitude. However, the magnetic sensor does not provide a reliably accurate measurement in general outdoor environments, since magnetic materials around and underneath a user can introduce severe errors. We thus propose an alternative for accurate yaw estimation using an aerial photograph instead of depending on the magnetometer alone.

The distance between a user and a reference building corner can be determined from the aerial photograph, for which we assume to possess scale information. Figure 2 illustrates a method for accurate yaw calculation on the basis of the aerial photograph. Our aerial photographs are all oriented with geographic north being aligned with

the photograph's y-axis. The angle ψ_B is easily obtained as the slope of line *ML*, which is established using the edge detection techniques from Section 3.3. The angle ψ_U can be calculated trigonometrically since the coordinates of the corner *M* and the user's position (U) coordinates are known. Then, the final yaw value ψ measuring the user's head orientation relative to the building orientation is calculated as follows:

$$\psi = \psi_B + (90^\circ - \psi_U) \quad (1)$$

Clearly, accurate corner detection is important in that it determines the preciseness of yaw.

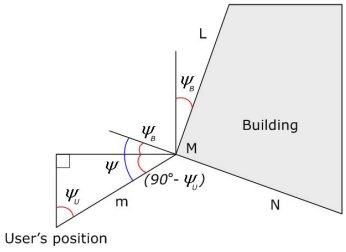


Figure 2: *Yaw calculation using an aerial photograph*

3.2 Height Estimation Using an Inertial Sensor

In order to generate a (partial) 3D model of the building in question, the height of the building has to be estimated. For this, we use our hybrid orientation tracker's pitch measurement, which is quite reliable because of its integrated inclinometer that corrects for gyroscopic drift.

While measuring pitch, it is assumed that a user is looking at a vertical edge of a building through a camera without a roll offset. To get pitch, the user first needs to look at one spot of a building edge which is the same height as the user's, calibrating the pitch to zero. In the next step, the user should look at the building corner in question by moving up his/her viewpoint along the vertical edge. For this calibration step, the building should be viewed vertically without a confounding influence of roll. If the pitch is calculated, then we can compute the height of the building using the following expression

$$h_B = h_U + m\tan(\theta) \quad (2)$$

where h_U denotes the head-mounted camera's height which is assumed to be known from the user's height, and m represents the ground distance between the user and the edge of the building as shown in Figure 5.

3.3 Accurate Edge and Corner Detection

This section describes how we detect edges and corners of a building in both the aerial photograph and the user's view. We reduced the need for user interaction to three mouse clicks that only have to lie within a certain search area around a true image corner. From these three clicks (two for the aerial photograph, one for the frontal view) the system automatically finds edges and corners in both views. The frontal view corner detection is currently implemented to work on a still image (frozen from the user's video stream), but will be adapted to work on the user's live camera view.

We use the OpenCV [Intel 2007] implementation of the Canny edge detector in both the aerial photograph (cropped to VGA resolution) and camera views (natively VGA). Selecting the best candidate edges in a robust fashion is non-trivial. Using the probabilistic Hough transform, we first find a set of lines with a minimum line length l as a threshold (l_{aerial}=10 pixels, $l_{frontal}$=30 pixels). As maximum gap, we chose 20 pixels and as the transform's accumulator value we chose 30 pixels. Figure 3 shows edge and corner detection results for a video frame. Figure 3(a) and Figure 3(b) depict the input image and Canny edge detection output. After applying the Hough transform, we get probabilistic Hough lines as shown in Figure 3(c). To extract only three edges around a building corner, lines passing through a search area of a certain size A around a user's input point are considered ($A_{aerial} = (11pixel)^2, A_{frontal} = (25pixel)^2$ – it makes sense for the frontal view search area to be larger since it will be harder to pinpoint a corner in a non-static image). The filtered lines are then classified into three groups according to slopes. A line of longest length is finally selected in each group. Figure 3(d) illustrates the final three edge lines and the corresponding corner for our example.

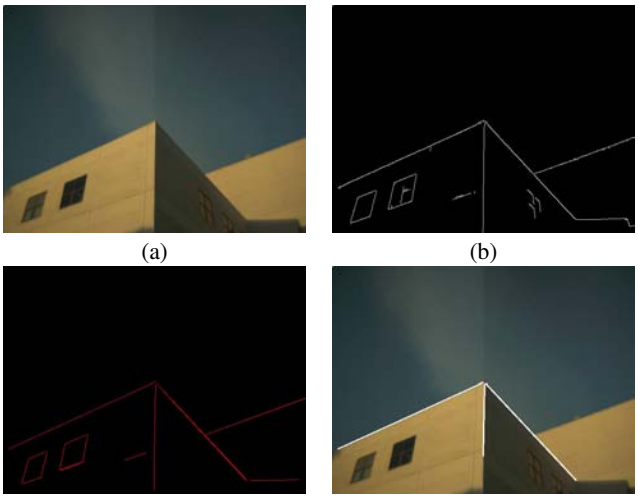


Figure 3: *Edge and corner detection in frontal view: (a) original image; (b) Canny edge detection results; (c) probabilistic Hough lines; (d) detected edges (white lines) and corner (red dot)*

The objective of adopting an aerial photograph is to estimate outlines and a corner of a target building. For an extraction of the outlines, the same method is employed except that we have to perform the procedure twice in the aerial photograph. Ideally, the coordinates of the edges of the rooftop and the bottom would be the same if the camera capturing the aerial photographs could be assumed as being orthogonal to the ground. However, as shown in Figure 4, there is a very noticeable building slant in the common case that the camera direction was not orthogonal to the ground. Thus, we extract separate edges for the rooftop and the bottom of the building. If the bottom of the building is not visible in the aerial photograph because of perspective occlusion, the user has to estimate the ground point based on their impression of the visible sides of the building in question, and a line set parallel to the L-shape on the roof through that estimated point is assumed. In general, we choose two lines that are likely to be parallel to the first line pair and orthogonal to each other, among the lines that pass within a search area around the user's input. The corners are estimated to be the intersection point of the two lines as shown in Figure 4. We can see the detection results for the rooftop and bottom parts in Figure 4(a) and Figure 4(b), respectively.

3.4 3D Model Generation

So far, we have discussed modeling box-shaped corners of buildings, which constitute parts of larger building structures in the physical world. If the overall building geometry is a simple box shape, the width and the depth of the building could be inferred from the aerial photograph and a whole building model can be computed and stored. If the building geometry is more complex, we anticipate to be able to piece together a complete building model by recognizing additional building corners while we are tracking the walls originating from a previous corner. This procedure will be similar to our current corner

(a) (b)

Figure 4: *Edge and corner detection in aerial photograph: (a) rooftop part; (b) ground part*

detection, but will be able to make use of additional constraints from the ongoing tracking and partially completed building representation. While we have begun first tests on this topic, this is clearly the scope of future work.

Figure 5 demonstrates a conceptual diagram for simple 3D model generation. *M* denotes a modeled corner of a building. *L* and *N* in Figure 5(c) are chosen, for convenience, to denote the directions of the building walls, not exact corners. Our basic assumption is that the building corner to be modeled is rather simple, such as in a box shape, without confounding sloped edges such as in more complicated rooftop structures.

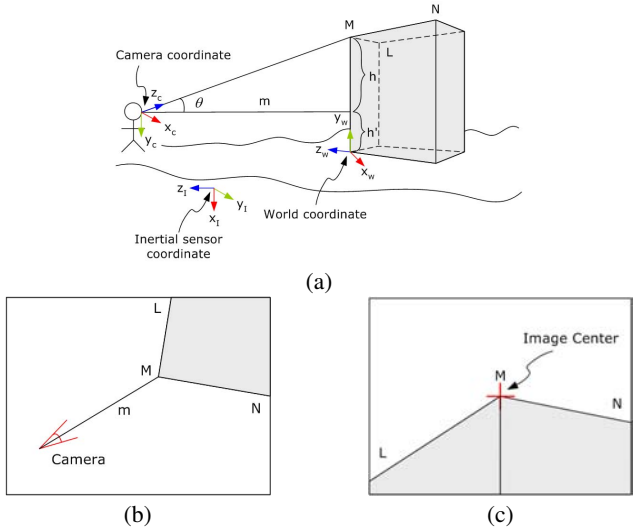


Figure 5: *A conceptual diagram for 3D model generation: (a) coordinate layout; (b) top-down view corresponding to aerial photograph; (c) frontal view corresponding to video image*

4 Camera Pose Tracking

This module tracks the pose of a camera mounted to an AR system in urban outdoor environments. The overall framework relies on feature-based tracking based on salient points on video frames. We filter the new pose estimates using a UKF with a constant velocity model.

4.1 Motion Model and System Dynamics

The proposed system relies on a 3D model of the scene to be tracked. Based on the prior estimate of a camera pose, a 3D model is projected into the camera's view for every frame, computing the visible parts of edges. A point $\mathbf{X} = (X, Y, Z, 1)^T$ in world coordinates is projected to the point $\mathbf{x} = (x, y, 1)^T$ using the camera projection matrix P as follows:

$$\mathbf{x} = P\mathbf{X} \tag{3}$$

$$P = K[R\,t], \quad K = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where $[R\ t]$ is a rigid-body transformation matrix, consisting of rotation and translation, mapping points from the world coordinate system into the camera coordinate system. They can be parameterized with a six-vector corresponding to translations and rotations around the three axes using the exponential map. f_x and f_y denote focal lengths along x and y axes, and x_0 and y_0 represent principal points along each axis, respectively. The projection of a 3D model will not correspond to salient points in the video frame due to the camera motion. The goal is then to compute a motion of the camera required to align the projection with the video image to provide a posterior pose estimate.

We employ a constant velocity model with 12 parameters: 6 parameters for camera pose and 6 parameters for velocity. The state vector at time T_k is as follows:

$$\mathrm{x}_k = (t_k, r_k, v_k, w_k)^T \tag{5}$$

where t_k and r_k denote translation and rotation vectors, respectively. On the other hand, v_k and ω_k represent velocity components for the corresponding vectors. For a small sampling period $\Delta T = T_k - T_{k-1}$, we can use a constant velocity model for the system dynamics as follows:

$$\mathrm{x}_k = \begin{bmatrix} t_k \\ r_k \\ v_k \\ w_k \end{bmatrix} = \begin{bmatrix} t_{k-1} + v_{k-1}\Delta T \\ r_{k-1} + w_{k-1}\Delta T \\ v_{k-1} + n_{v,k-1} \\ w_{k-1} + n_{w,k-1} \end{bmatrix} \tag{6}$$

where n_v and n_w are the random distribution noise components of translation velocity and instantaneous rotation velocity, respectively.

4.2 Salient Point Detection

Offline intrinsic camera calibration is done using Zhang's procedure [2000]. The distortion coefficients from this process are used to correct the resulting artifacts in the video images by creating a corresponding undistortion image warp that is applied to each frame. Features on the building walls are selected in video frames by Shi and Tomasi's algorithm [1994], which finds a set of all features of a certain quality and then greedily selects features from the set that are not within a minimum distance of the already selected features. These features are transformed to control points in 3D space through the following back-projection. Given a point $\mathbf{x}$ in an image, we can determine the set of points in space that map to this point. This set constitutes a ray in space passing through the camera center. Writing the camera projection matrix $P = [M|p_4]$, the camera center is given by $\widetilde{C} = -M^{-1}p_4$ [Hartley and Zisserman 2004]. An image point $\mathbf{x}$ back-projects to a ray intersecting the plane at infinity at the point $D = ((M^{-1}\mathbf{x})^T, 0)^T$, and D provides a second point on the ray. We can thus write the line as the join of two points on the ray as follows.

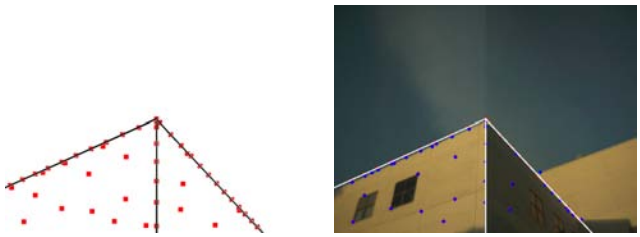
$$\begin{aligned} X(\mu) &= \mu \begin{pmatrix} M^{-1}\mathrm{x} \\ 0 \end{pmatrix} + \begin{pmatrix} -M^{-1}p_4 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} M^{-1}(\mu\mathrm{x} - p_4) \\ 1 \end{pmatrix} \end{aligned} \tag{7}$$

Thus, μ is the only remaining parameter to be determined. However, since equations of the planes corresponding to the building walls are already known, we can calculate the intersection points between each ray from a point $\mathbf{x}$ in a frame and each wall plane of the virtual 3D

building. Figure 6(a) demonstrates the control points on the sides of a building as well as on the edges.

The corresponding salient points on the sides of a building are tracked frame to frame using the image-pyramid-based optical flow algorithm of Lucas and Kanade [1981]. A hierarchy of images at different resolutions are used to efficiently match texture features from one frame with the most similar region in another frame. If the similarity between these two regions is below a threshold, the feature is considered lost and is removed from the set. This can happen when a feature goes outside the field of view, or when changes in illumination or occlusion occur.

On the other hand, salient points on the edges are detected by comparing edges found in the video feed with control points on edges rendered using the camera pose estimate. We use a 3D OpenGL model containing the control points of the object to be tracked. Next, the control points are initialized at regular intervals along every 3D model edge. Each control point is projected and rendered to screen after visibility test against the z-buffer. At every visible control point, one-dimensional edge detection in the direction of the model's edge normal is performed in the video image. The closest intensity edge detected within a cutoff distance of the control point is assumed to be the salient point of the video image edge corresponding to the current model edge. The coordinates of these salient points on the detected edge are fed as a measurement into the UKF. Figure 6 shows salient point detection results on edges and sides of a building. Control points and salient points are depicted as red dots and blue dots, respectively, and the distance between them is depicted as a green line in Figure 6(b)



(a) (b)

Figure 6: *Salient point detection on edges and sides of a building (a) control points on a generated 3D model (b) corresponding salient points on a video image*

4.3 Pose Estimation Using the UFK

The frame-to-frame tracking of the generated model in 3D space by comparing it to video images acquired from a camera can be formulated as a nonlinear estimation problem. In particular, the UKF often produces better estimates of the covariance matrices of the parameters involved compared to the Extended Kalman Filter (EKF) [Haykin 2001]. It is also more efficient and simpler to implement, avoiding the computation of a Jacobian matrix, which is necessary to propagate distributions in the EKF. Instead, a small number of carefully chosen sample points are propagated in each estimation step, which provides a compact parametrization of the underlying distribution.

First, we define the state vector $\mathbf{x}_k$ as shown in Eq. (5), and the equivalent process model is expressed as follows

$$\mathrm{x}_k = F\mathrm{x}_{k-1} + w_k \tag{8}$$

where the state transition matrix F, derived from Eq. (6), can be defined as follows: w_k is the Gaussian process noise with covariance matrix Q and is determined empirically based on the observed measurements.

$$F = \begin{bmatrix} I_6 & (\Delta T)I_6 \\ 0 & I_6 \end{bmatrix} \tag{9}$$

where I is an identity matrix, and ΔT is the time period between captured frames. Thus, for the linear process model, the time update equations are as follows:

$$\widehat{\mathrm{x}}_{k|k-1} = F\widehat{\mathrm{x}}_{k-1|k-1} \tag{10}$$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q \tag{11}$$

where $P_{k|k}$ is the filter's state error covariance at time T_k and Q denotes the covariance matrix of the process noise w_k.

Based on the unscented transform (UT) from [Wan and Merwe 2000] and [Julier and Uhlmann 2004], $2L+1$ sigma points are generated as follows.

$$\chi = \begin{bmatrix} \widehat{\mathrm{x}}_{k|k-1} & \widehat{\mathrm{x}}_{k|k-1} + \gamma\sqrt{P_{k|k-1}} & \widehat{\mathrm{x}}_{k|k-1} - \gamma\sqrt{P_{k|k-1}} \end{bmatrix}$$
$$\gamma = \sqrt{L+\lambda}, \quad \lambda = \alpha^2(L+\kappa) - L \tag{12}$$

where the initial values of $\widehat{\mathrm{x}}_{k|k-1}$ and $P_{k|k\text{-}1}$ are determined experimentally. $L = 12$ is the number of elements in the state vector, and $\sqrt{P_{k|k\text{-}1}}$ is the Cholesky decomposition of the predicted covariance. The constant α determines the spread of the sigma points around $\widehat{\mathrm{x}}_{k|k-1}$, and the constant κ is a secondary scaling parameter, which is set to $3 - L$.

Even though the measurement function is based on Eq. (3), since we are dealing with N_S salient points, which may vary at every frame, we can rewrite as follows:

$$\begin{bmatrix} x_k^i \\ y_k^i \end{bmatrix} = \begin{pmatrix} f_x \frac{R_1(r_k)\cdot[X-t_k]}{R_3(r_k)\cdot[X-t_k]} + x_0 + n_x \\ f_y \frac{R_2(r_k)\cdot[X-t_k]}{R_3(r_k)\cdot[X-t_k]} + y_0 + n_y \end{pmatrix}$$
$$i = 0, ..., N_s - 1 \tag{13}$$

where R_j is the j^{th} row vector of rotation matrix $R(r_k)$. nx and ny model the measurement noise of feature detection, while i denotes the i^{th} feature from 0 up to $N_S - 1$. In summary, the control points shown in Figure 6 are projected onto the image plane based on the sigma points of Eq. (12) through the measurement function Eq. (13) and then the UFK updates the camera pose estimates by minimizing the distance from the estimated points to the salient points, which are obtained as in Section 4.2.

5 Results

The experiments were performed on our wearable system. At its core is an IBM T43 laptop computer with a 2.0 GHz CPU, which is worn on the user's back. The display is an SVGA Sony Glasstron PLM-S700E attached to the front of a helmet, used in a video see-through mode. Mounted directly above the display are a PointGrey Dragonfly2 IEEE 1394 camera and an InterSense InertiaCube2 orientation tracker, and on top of the helmet is a Garmin GPS 18 position tracker. User input is through a hand-held ErgoTouch RocketMouse. All of these devices are relatively inexpensive, off-the-shelf components.

The intrinsic camera parameters were evaluated by Zhang's calibration procedure using the implementation of OpenCV [Intel 2007]. Parameters for the UKF were set as follows; $L = 12$, $\alpha = 10^{-4}$, $\beta = 2$, $\kappa = -9$.

We implemented a prototype application that lets a user model a suitable building and introduce live annotations on its facade. While walking alongside the building, a user can follow how well the initial partial building model stays registered with the physical world and can introduce new annotations (cf. labels "Jason's Room" and "Steve's Room" in Figure 7.) The user only needed to click onto any pixel occupied by the building projection, at any time, and the system calculated the corresponding 3D points on the walls, using the back-projection technique on the clicked 2D image points through Eq. (7). We implemented two modes of label stabilization: The first one simply treats the annotation anchors as part of the tracked 3D building

model, while the second one keeps the anchor stable with regard to the nearest 2D control point (which results in more stable local registration).

Figure 7: *Live annotation during tracking*

5.1 Accuracy

To evaluate the accuracy of the localization, we compared our results to ground truth data delineated on the aerial photograph. For a 3D point (x, y, z), the coordinates x and z correspond to the easting and the northing of the aerial photograph, while y corresponds to elevation above the ground plane. Figure 8 shows camera coordinates (x, z) for a sequence where the camera was set on a tripod above a known point during the pose estimation, facing the modeled building-part. The standard deviations of the estimated camera positions are (sx, sy, sz) = (0.0137 m, 0.0165 m, 0.01323 m). The mean $(\widehat{x}, \widehat{y}, \widehat{z})$ = (-7.3755 m, 1.0438m, -18.6822 m) has a small discrepancy from the physically measured point (-7.4690 m, 1.25 m, -18.7493 m), resulting from calibration and sensing errors.

In another type of experiment, to evaluate the accuracy under motion, we let a user walk along a marked trajectory (the outer edge of a bike lane, cf. Figure 9(a) and dotted line in Figure 9(b)), and recorded the estimated camera poses. Figure 9(c) demonstrates the distance error of the estimated camera positions to the bike lane edge over time using the UKF against a least squares (LS) estimator. We can observe that the UKF implementation produces smaller errors than the LS estimator. It is also worthwhile to note that the LS estimator produces high peaks in comparison to the UKF, meaning that the former is vulnerable to fast motion and sensitive to noise.

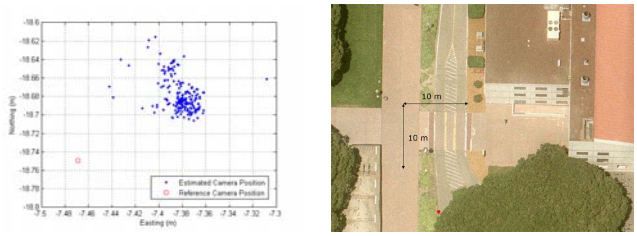
Table 1 shows comparison results of the UKF against the LS estimator in terms of the mean and standard deviation of the distance errors to the ground truth line, and the average processing time. The results demonstrate that the average accuracy of the UKF is better than that of the LS estimator even though the former requires a little more average processing time than the latter. Note, however, that while the processing time of the UKF is sufficient for real-time applications, the peaks of the LS estimator can have a critical effect on tracking.

Table 1: *Comparison between the LS estimator and the UKF*

Method	Average RMS error (m)	Standard deviation (m)	Average processing time (ms)
LS	0.265	0.182	9.712
UKF	0.245	0.116	16.636

5.2 Performance

The system currently operates at about 14-17 frames per second. Table 2 gives an overview of the average processing times of individual steps. The total average operating rate for tracking is at approximately 25 Hz. The DragonFly2 camera, however, uses API functions for frame grabbing and preprocessing which reduce the system frame rate substantially. Consequently, while the tracking operation alone can run at about 25 Hz, other processing steps such as acquiring an input frame and preprocessing incur sufficient overhead to lower the system frame rate to 14-17 Hz.



(a) (b)

Figure 8: *Tracking accuracy for a stationary camera: (a) distribution of estimated camera coordinates (x, z) for a stationary pose. Circle denotes reference point from aerial photograph (b) estimated camera positions (red dots) on the aerial photograph*

Table 2: *Average processing times for individual steps*

Individual Step	Processing Time (ms)
Feature tracking on the walls	21.810
Edgel extraction	0.236
Salient point search	1.283
Unscented Kalman Filtering	16.636
Total	39.965

5.3 Analysis and Discussion

Initial localization (via a GPS and manual user correction using the aerial photograph) has a large effect on subsequent steps since the initial projection of a 3D model onto an image plane is determined based on pitch and yaw which are calculated from the initial user position. The effects of error in the initial localization step are depicted in Figure 10. Note that pitch and yaw are a function of the user's position. In this paper, we do not analyze this relationship in detail. Instead, we investigate the influence of the pitch and yaw variations on 3D model generation and tracking.

Let us first consider the distance variation case where we assume that the position estimated through the GPS has an error towards either m_{far} or m_{near} direction. However, since the user is looking at the corner of the building and only the *estimated* position varies, not the true position, we can see that the reference pitch $\theta_{ref} = \theta_{near} = \theta_{far}$, and only m varies as illustrated in Figure 10(b). Thus, the height estimated based on Eq. (2) is different from the real one, and the results are shown in Figure 10(c). The yaw variation has also a similar effect on the initialization procedure. According to the user's position towards ψ_R or ψ_L direction, the rendered 3D building model is skewed as shown in Figure 10(d). Unless the estimation is sufficiently accurate in both cases, the distances from control points to building edges on a video image may exceed the length of the search range, l_{sr}. This would result in tracking failures.

Let us provide an example where m_{ref} = 19.579 m, $h_B - h_U$ = 14.225 m, $\theta_{ref} = 36.0^\circ$, $\psi_U = 68.1^\circ$, and l_{sr} = 40 pixels. In this case, for the control points to meet the corresponding salient points, the yaw error, $\theta_{ref} - \theta_L$ or $\theta_R - \theta_{ref}$ should not exceed about 4°. Similarly, distance error $m_{ref} - m_{near}$ or $m_{far} - m_{ref}$ should not exceed about 1m.

Through the experiments, we observed that the 3D model generated through the proposed methodology was sufficiently accurate to be used in the tracking framework. One of the most essential factors to be considered for the modeling is a user's initial position, estimated mainly through the GPS unit. Since the off-the-shelf non-differential GPS was employed, we compensated the inherent errors using a high-resolution aerial photograph and user intervention. Differential or

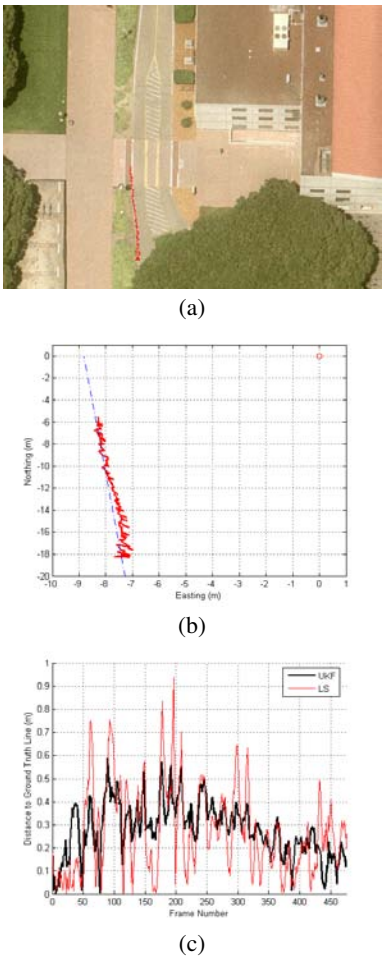


Figure 9: *A trajectory on an aerial photograph as produced by the pose tracking: (a) tracking results on aerial photograph; (b) tracking results against ground truth (dotted line). Circle at origin (0, 0) represents bottom corner of building in a) (c) comparison plot of UKF against LS estimator results*

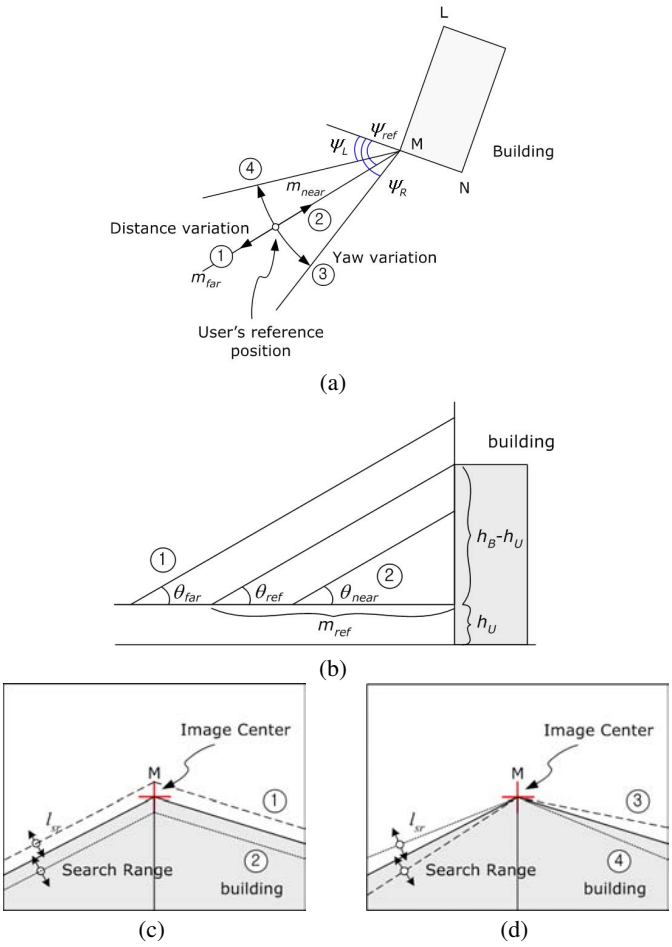


Figure 10: *Error analysis for initial user localization (a) overhead view illustrating possible positions with yaw or distance error (b) distance variation inducing error in measured height (c) projected building silhouette in presence of distance deviation (d) projected building silhouette in presence of yaw deviation*

WAAS GPS systems would provide a better starting estimate, but are not in agreement with the goals of Anywhere Augmentation. We also observed that although the InertiaCube2 was heavily affected by magnetic material around a user, we could get fairly accurate pitch after turning off the compass functionality of the InertiaCube2. In addition, since the resolution of the aerial photograph set that we are using is 0.0762 m per pixel, the yaw estimation was accurate enough to coincide the 3D model with the video image. Our aerial photograph set was highly non-orthogonal. That was the reason we had to find two corners at the rooftop and the bottom of a building separately. However, when high-resolution satellite images or orthogonal aerial photographs become available, the user interaction for the 3D modeling procedure will be even more simplified.

In our initial tracking, which was based on model edges alone, we experienced a shrinking/expanding problem, which occurred in the case of tracking a large building, since at most one corner and two sides of a building can be viewed within a single video image. We encountered a problem when the number of used features became too small. Thus, we included the salient features on the side walls to prevent the model from being shrunk or expanded. However, with a large number of salient points, the time spent in the UKF increases. Thus, the number of total salient points should be balanced properly.

Finally, the detection issue of edges and features on a building during the tracking is very critical. As is true for most computer vision systems, the edge and feature detection is affected by lighting conditions, especially by sunlight in outdoor environments. When sunlight was too strong around noon, salient features on the sides of a building were hardly detected. In this case, we controlled two camera parameters, Gamma and Shutter, so that we could avoid color saturation in our video frames. On the other hand, we occasionally ran into edge detection problem when the colors of the sky and a building were too similar. When the boundary is not clear enough for edge detection, the probabilistic hough transform had nothing to work with. However, high pass filters, such as the High-boost filter or the Homomorphic filter can be employed, so that the boundary between the sky and a building becomes more emphasized [Gonzalez and Woods 2002]. In our case, we did not use such a filter due to processing time restrictions.

In order to maintain the registration of our partial model with the physical building it describes, the user has to keep the modeled building corner in view. This is acceptable for very simple prototype applications, in which a user simply wants to add an annotation to an object and check it before committing it to a global database, but for general Anywhere Augmentation, we clearly need to address the robustness of the system. The user should be enabled to look away from the (or any) building and re-calibrate automatically when looking back at it. This can be addressed by future work on incorporating persistent scale-independent landmark features. Possible occlusion of modeled

geometry by foreground objects (such as trees) has to be addressed as well.

6 Conclusions and Future Work

We presented an online 3D AR modeling method that uses information from aerial photographs and simple user interaction in conjunction with image processing and computer-vision tracking to create a simple model of part of a building. We then enabled the system to track the 3D model using salient points on both the edges and sides of a building, using a UKF framework. Our results demonstrate that a user can create a 3D model on the fly with enough accuracy for tracking, and that this 3D model can be tracked in real-time for outdoor AR applications. There are still several remaining challenges. The current off-the-shelf GPS unit provides a basic, but inaccurate estimate of a user's position. For initialization of the position, the accuracy of the GPS unit must be improved by combining with other sensors. In addition, we are currently working on integrating the output from our inertial tracker with the vision-based tracker using the UKF, in order to cope with fast motion. Instead of our current UKF implementation that uses $2L + 1$ sigma points, we are exploring a $L + 2$ sigma point approach, in the hope to improve tracking performance with respect to computational complexity as well as accuracy. Furthermore, we are going beyond simple cube-shaped buildings. This can be done by incorporating more SLAM aspects into our algorithm, allowing us to track scenes consisting of non-connected objects, and by joining new building corners with existing partial models. Corner models themselves should be generalized so that we may deal with buildings of arbitrary rooftop and facade shapes. Finally, we are working on approaches to make the system robust to obstacles such as trees, light poles, passing people, or cars.

References

AZUMA, R., BAILLOT, Y., BEHRINGER, R., FEINER, S., JULIER, S., AND MACINTYRE, B. 2001. Recent advances in augmented reality. *IEEE Computer Graphics & Application 21*, 6, 34–47.

BAILLOT, Y., BROWN, D., AND JULIER, S. 2001. Authoring of physical models using mobile computers. In *Proceedings of ISWC'01*, 39–46.

BEHRINGER, R. 1999. Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors. In *Proceedings of IEEE VR'99*, 244–251.

COORS, V., HUCH, T., AND KRETSCHMER, U. 1999. Matching buildings: Pose estimation in an urban environment. In *Proceedings of ISAR'00*, 89–92.

DAVISON, A., MAYOL, W., AND MURRAY, D. 2003. Real-time localisation and mapping with wearable active vision. In *ISMAR '03: Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, Washington, DC, USA, 18.

DAVISON, A. 2003. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision*, 1403–1410.

DRUMMOND, T., AND CIPOLLA, R. 1999. Visual tracking and control using lie algebras. In *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition'99*, 652–657.

FEINER, S., MACINTYRE, B., HÖLLERER, T., AND WEBSTER, A. 1997. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban enviroment. In *Proceedings of ISWC'97*, 74–81.

GONZALEZ, R., AND WOODS, R. 2002. *Digital Image Processing*. Prentice Hall.

GOOGLE, 2007. Google maps. http://maps.google.com/, May.

HARTLEY, R., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

HAYKIN, S. 2001. *Kalman Filtering and Neural Networks*. John Wiley & Sons, Inc.

HOFMANN-WELLENHOF, B., LICHTENEGGER, H., AND COLLINS, J. 1997. *Global Positioning System: Theory and Practice*. Springer.

HÖLLERER, T., WITHER, J., AND DIVERDI, S. 2007. *Anywhere Augmentation: Towards Mobile Augmented Reality in Unprepared Environments*. G. Gartner, M.P. Peterson, and W. Cartwright (Eds.), Location Based Services and TeleCartography, Series: Lecture Notes in Geoinformation and Cartography, Springer Verlag.

INTEL, 2007. Open source computer vision library. http://www.intel.com/technology/computing/opencv/, May.

JULIER, S., AND UHLMANN, J. 2004. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, vol. 92, 401–422.

KING, G., PIEKARSKI, W., AND THOMAS, B. 2005. Arvino - outdoor augmented reality visualisation of viticulture gis data. In *Proceedings of IEEE ISMAR'05*, 52–55.

KLEIN, G., AND DRUMMOND, T. 2003. Robust visual tracking for noninstrumented augmented reality. In *Proceedings of IEEE ISMAR'03*, 113–122.

LUCAS, B., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 674–679.

PIEKARSKI, W., AND THOMAS, B. 2001. Tinmith-Metro: New outdoor techniques for creating city models with an augmented reality wearable computer. In *Proc. ISWC '01 (Fifth Int. Symp. on Wearable Computers)*, 31–38.

PTGREY, 2007. Point grey research inc. http://www.ptgrey.com/, May.

REITMAYR, G., AND DRUMMOND, T. 2006. Going out: Robust model-based tracking for outdoor augmented reality. In *Proceedings of IEEE ISMAR'06*, 109–118.

RIBO, M., LANG, P., GANSTER, H., BRANDNER, M., STOCK, C., AND PINZ, A. 2002. Hybrid tracking for outdoor augmented reality applications. *IEEE Comp. Graph. Appl. 22*, 6, 54–63.

ROSTEN, E., AND DRUMMOND, T. 2005. Fusing points and lines for high performance tracking. In *Proceedings of ICCV'05*, 1508–1511.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *Proceedings of CVPR'94*, 593–600.

SIMON, G., FITZGIBBON, A., AND ZISSERMAN, A. 2000. Markerless tracking using planar structures in the scene. In *Proceedings of IEEE and ACM ISAR'00*, 120–128.

VACCHETTI, L., LEPETIT, V., AND FUA, P. 2004. Combining edge and texture information for real-time accurate 3d camera tracking. In *Proceedings of ISMAR'04*, 48–57.

WAN, E., AND MERWE, R. V. D. 2000. Unscented filtering and nonlinear estimation. In *Proceedings of Adaptive Systems for Signal Processing, Communications, and Control Symposium '00*, 153–158.

WITHER, J., DIVERDI, S., AND HÖLLERER, T. 2006. Using aerial photographs for improved mobile ar annotation. In *Proceedings of IEEE ISMAR'06*, 159–162.

YAHOO, 2007. Yahoo maps. http://maps.yahoo.com/, May.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *Transactions on PAMI 22*, 11, 1330–1334.