Vision-Based Tracking for Mobile Augmented Reality

Fakhr-Eddine Ababsa, Madjid Maidi, Jean-Yves Didier, Malik Mallem

Vision-Based Tracking for Mobile Augmented Reality

Fakhreddine Ababsa, Madjid Maidi, Jean-Yves Didier, and Malik Mallem

IBISC Laboratory CNRS FRE 2873. 40 rue du Pelvoux 91020 Evry, France
ababsa@iup.univ-evry.fr, maidi@iup.univ-evry.fr
didier@iup.univ-evry.fr, mallem@iup.univ-evry.fr

Summary. Augmented Reality Systems (ARS) attempt to enhance humans' perception of their indoors and outdoors working and living environments and understanding of tasks that they need to carry out. The enhancement is effected by complementing the human senses with virtual input. For example, when the human visual sense is enhanced, an ARS allows virtual objects to be superimposed on a real world by projecting the virtual objects onto real objects. This provides the human user of the ARS with additional information that he/she could not perceive with his/her senses. In order to receive the virtual input and sense the world around them augmented with real time computer-generated features, users of an ARS need to wear special equipment, such as head-mounted devices or wearable computing gears. Tracking technologies are very important in an ARS and, in fact, constitute one challenging research and development topic. Tracking technologies involve both hardware and software issues, but in this chapter we focus on tracking computation. Tracking computation refers to the problem of estimating the position and orientation of the ARS user's viewpoint, assuming the user to carry a wearable camera. Tracking computation is crucial in order to display the composed images properly and maintain correct registration of real and virtual worlds. This tracking problem has recently become a highly active area of research in ARS. Indeed, in recent years, several approaches to vision-based tracking using a wearable camera have been proposed, that can be classified into two main categories, namely "marker-based tracking" and "marker-less tracking." In this chapter, we provide a concise introduction to vision-based tracking for mobile ARS and present an overview of the most popular approaches recently developed in this research area. We also present several practical examples illustrating how to conceive and to evaluate such systems.

12.1 Problem Formulation

In vision-based tracking approaches, image features considered for pose computation are often points, lines, contours or a combination of these different features. To illustrate the formalism of the pose estimation problem, we consider the case of point features.

Let $\mathbf{p}_i = (x_i,\ y_i,\ z_i)^t,\ i = 1, \ldots, n,\ n \geq 3$ be a set of 3-D non-collinear reference points defined in the world reference frame. The corresponding camera-space coordinates $\mathbf{q}_i = (x_i',\ y_i',\ z_i')$ are given by:

$$\mathbf{q}_i = R\mathbf{p}_i + T, \tag{12.1}$$

where $R = (\mathbf{r}_1^t, \mathbf{r}_2^t, \mathbf{r}_3^t)^t$ and $T = (t_x, t_y, t_z)^t$ are a rotation matrix and a translation vector, respectively. R and T describe the rigid body transformation from the world coordinate system to the camera coordinate system and are precisely the parameters associated with the camera pose problem. We assume that internal calibration parameters of the camera, such as focal length, principal point, lens distortion, etc. are known.

Let the image point $\mathbf{m}_i = (u_i,\ v_i,\ 1)^t$ be the perspective projection of $\mathbf{p}_i$ on the normalized image plane, as in Fig. 12.1. Using the camera pinhole model, the relationship between $\mathbf{m}_i$ and $\mathbf{p}_i$ is given by:

$$\mathbf{m}_i = \frac{1}{\mathbf{r}_3^t \mathbf{p}_i + t_z} \left(R\mathbf{p}_i + T \right). \tag{12.2}$$

Equation (12.2) is known as the *collinearity equation* and indicates that $\mathbf{m}_i,\ \mathbf{q}_i$ and the projection center of the camera O are collinear.

The pose estimation problem can be stated as that of finding R and T that minimize the re-projection error between the observed 2-D image points and the forward-projection of the known 3-D object points:

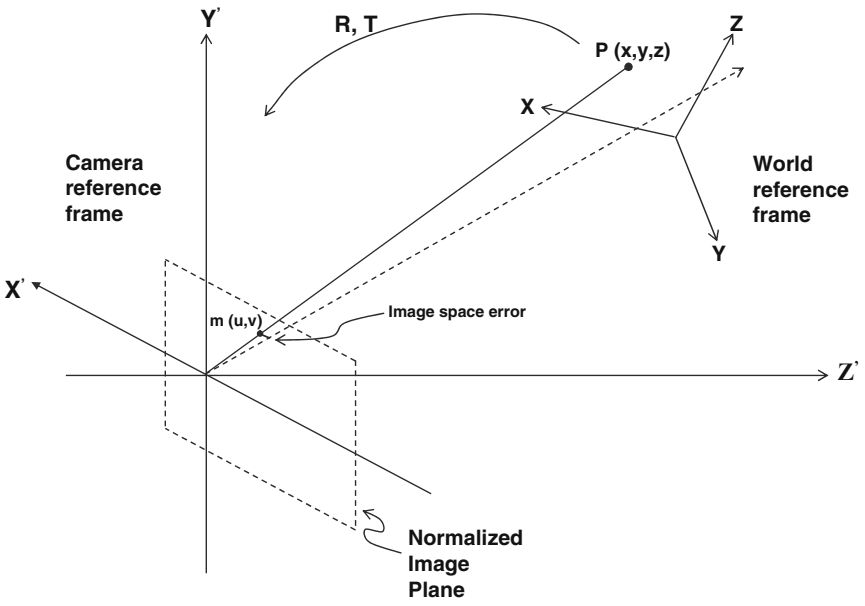

Fig. 12.1. Point constraints for the camera pose problem

$$\mathrm{E}\,(R,T) = \sum_{j} \left\| \mathbf{m}_i - \frac{(R\mathbf{p}_i + T)}{\mathbf{r}_3^t \mathbf{p}_i + t_z} \right\|^2 . \tag{12.3}$$

Numerical nonlinear optimization techniques, such as the Newton-Raphson or Levenberg-Marquardt algorithm, can be used to achieve the minimization. Also, there exist several iterative solutions based on minimizing the error $E(R,T)$ under certain nonlinear geometric constraints. Typical of these approaches is the work of Lowe [1] and Haralick [2]. Dementhon and Davis [3] initialize their iterative scheme (named POSIT) by relaxing the camera model to scaled orthographic. Their scheme uses at least four non-coplanar points. Lu et al. [4] reformulate the pose estimation problem as that of minimizing an object-space collinearity error. They combine a constraint on the world points, effectively incorporating depth, with an optimal update step in the iteration.

Purely geometric approaches have also been developed to solve the camera pose estimation problem. Their aim is to recover the camera pose relative to the scene object using geometric constraints. For example, Hung et al. [5] have proposed a method for fiducial pose estimation using four non-aligned and coplanar points. Quan and Lan [6] propose a family of linear methods that yield a unique solution to four- and five-points pose determination for generic reference points.

The basic idea of the camera pose estimation methods is to find correspondences between 2-D image features and their 3-D coordinates in a definite world frame. Marker-based approaches identify fiducials in the images and then extract 2-D interesting points from the markers regions. Whereas, marker-less based approaches extract directly interesting image features which correspond to the natural features in the 3-D environment. The 2-D-to-3-D correspondences are then obtained by using line and contour tracking approaches. In the following sections we will give more details on these two approaches.

12.2 Marker-based Approaches

To estimate the camera pose, it is necessary to have a set of 2-D points and their 3-D counter parts. These 2-D-to-3-D matchings are determined after detecting and identifying the object of interest in the image. One way to solve the pose estimation problem in real time is to resort to target (also called fiducials or features) extraction and tracking. These targets are stuck to the objects in a scene. Assuming the relative spatial transformation between the target and the object known and invariant, we are able to determine the position and orientation of the camera relative to the object. Usually, fiducials are embedded with a code that allows us to distinguish between several targets and track multiple objects in the same scene. Using the codes, we can then establish a semantic link between the tracking application and the

objects in the scene. Indeed, the target extraction is a known problem in computer vision and many existing systems rely on fiducials as we will see in the next section. Also, we will explore a case study detailing each stage of these classes of techniques, from image processing operators to the final pose estimation.

12.2.1 Related Works on Fiducial Extraction

Augmented Reality (AR) applications in the past years rely on fiducial extraction techniques to solve the pose estimation problem in real time. One of the requirements of fiducials is that they should possess simple geometrical shapes that can be easily detected with very fast and basic image processing filters. Amongst the numerous fiducial systems, two classes of shapes are commonly used, namely squares and circles.

Cho and Neumann [7] relied on a set of fiducials using multiple concentric colored rings. Colored areas are detected by expanding candidate pixels compared against reference colors. A centroid for the feature is computed by weighting the pixels with their distance from the reference color. The value of this centroid will give one 2-D point for each target. Since the camera is calibrated and the positions of the markers are known, at least three fiducials are needed to estimate the pose of the camera.

Naimark and Foxlin [8] developed their own system of coded targets. Their targets were composed of circular shapes. The algorithm of target recognition consists of four steps, namely contrast enhancement, edge detection, binarization and erosion of the image. The target code is read using white points in the target center and two black spots located in the target quadrants. This set of three points forms a reference frame to extract the code. Using circular targets, implies that it is necessary to have at least three fiducials to compute the camera pose. It also implies that several targets should be placed on the same object in order to track it. This is the reason why some systems are based on square targets, the four corners of which provide directly sufficient points for pose estimation.

Rekimoto [9] developed a localization system using a single target in the image. The system was initially called Matrix, but later it was renamed Cybercode [10]. This method is composed of several steps to detect targets and to estimate the camera pose. First, the image is binarized and an analysis of connected components is performed to determine black zones of the image and find the bar located under the code. Then, from the bar localization, the four corners of the code are found and finally the image is normalized in order to extract the code composed of 33 bits.

Kato and Billinghurst [11] designed the ARToolkit library for the rapid development of AR applications. This library provides computer vision techniques to compute the position and orientation of a camera relative to marked targets so that virtual 3-D objects can be overlaid on the markers. First the

(a) Cybercode (b) Intersense (c) ARToolkit

Fig. 12.2. Some examples of different coded fiducials

image is converted into a binary image based on a lighting threshold value. ARToolKit finds all squares in the binary image and, for each square, captures the pattern inside matches to some pretrained pattern templates. If there is a match, then ARToolKit has found one of the AR tracking markers. Finally, computer graphics are drawn over the real marker.

Besides the previous, other systems, were developed by several laboratories, such as the four compared in Zhang [12]. Moreover, recent work has been performed on increasing the robustness of fiducial recognition and code extraction [13, 14].

Some of the evoked coded fiducial can be seen in Fig. 12.2. To illustrate the fiducial-based technique, we will detail next our fiducial extraction and identification approaches as they have been tailored to fit our needs. Specifically, we will evaluate pose estimation algorithms using one or several markers.

12.2.2 Square Fiducial Identification Approach

To extract the target from images, it is necessary to detect the object shape before identification. In order to reduce detection error rates, images are preprocessed into an acceptable form before carrying out any image analysis. The image is converted into a black and white image using a suitable threshold. Then, several operations are applied to process the image and detect the object shape. The algorithm of object detection is composed of the following steps, as in Fig. 12.3:

1. Apply Canny filter [15] to detect contours in image (Fig. 3.1)
2. Smooth the image using a Gaussian filter (Fig. 3.2)
3. Dilate the image to remove potential holes between segments (Fig. 3.3)
4. Make a polygonal approximation of contours and discard the ones that are not quadrilaterals (Fig. 3.4).

Once a potentially square object is detected, the next step is to identify this object and match it with a defined template by extracting a digital code. This is computed by mapping a set of reference points from the model of the fiducial to the actual image of the target which has undergone through a

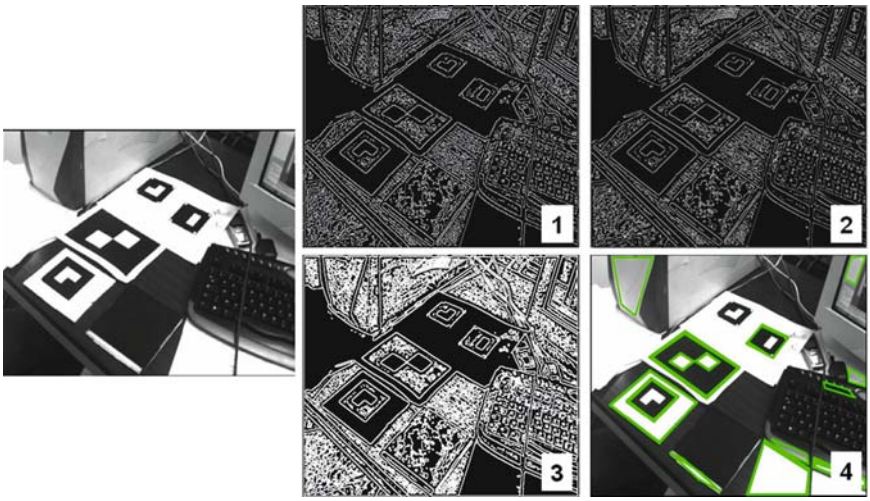

Fig. 12.3. Fiducial detection process

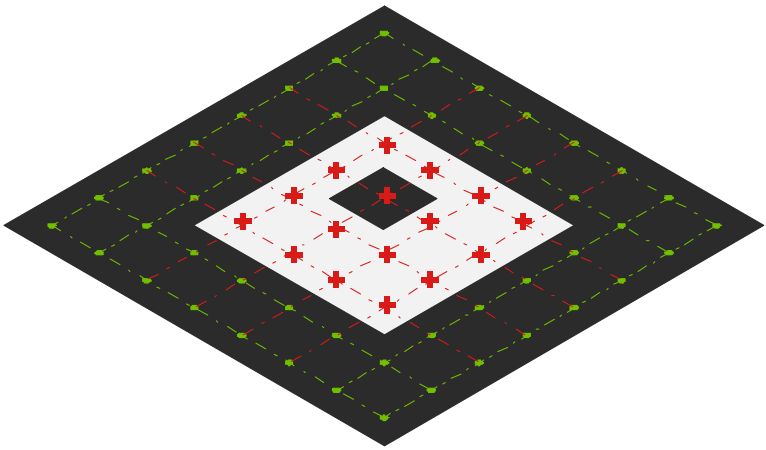

Fig. 12.4. Fiducial sampling

spatial transformation and projection. This mapping is performed by solving the following homography equation:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{12.4}$$

with (x, y) being the coordinates of the reference point and (u, v) the coordinates of the same point in the image. By arbitrarily setting $h_{33} = 1$, we can rewrite (12.4) as:

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -xu & -yu \\ 0 & 0 & 0 & x & y & 1 & -xv & -yv \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} h_{33}u \\ h_{33}v \end{pmatrix}. \tag{12.5}$$

The number of parameters to estimate is 8 and, thus, we need eight equations to solve. Therefore, we use four coplanar points which represent the fiducial vertices in the image. The set of reference points is a sampling grid (see Fig. 12.4) applied using the computed homography. Points in green should be black, otherwise the pattern is rejected because it must have a black border. The sixteen points in red are sampled to extract the corresponding code.

Our code is then composed of 16 bits and allows $2^{16} = 65{,}536$ possible different targets. However, the target system must respect a strong constraint, namely it should allow the detection of the target orientation. Each target which has been rotated by 90° has a different code in the identification phase. Thus, targets have four codes following their orientations and, consequently, the number of target classes is divided by 4. This reduces the number of possible codes, as in Fig. 12.5. Moreover, targets should not have a central symmetry because in that case target orientation cannot be determined. Finally, one obtains 16,320 classes of code and each target has a code from 0 to 65,535 and a unique orientation in object space.

12.2.3 Pose Estimation from Fiducials

Once the fiducial is recognized, we can proceed to pose estimation. To determine the pose, the 2-D-to-3-D pairs of points must be known and the camera assumed calibrated. The camera calibration determines the geometrical model

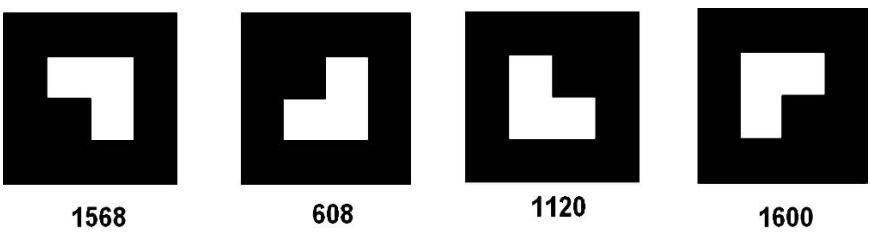

Fig. 12.5. Codes corresponding to different target orientation

of an object and the corresponding image formation system which is described by the following equation [16]:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{12.6}$$

In(6), s is an arbitrary scale factor and $(R,\ t)$, called the *extrinsic* parameters, are the rotation and translation that transform the world coordinate system to the camera coordinate system. The remaining parameters, called *intrinsic* parameters, are the coordinates $(u_0,\ v_0)$ of the principal point and the scale factors $\alpha_u,\ \alpha_v$ along the u and v image axes. The intrinsic parameters are computed during the camera calibration procedure and remain unchanged throughout the experiments.

The aim of pose estimation is to compute the extrinsic matrix $(R,\ T)$ using couples of 2-D-to-3-D matched points, in this case corners of our square fiducial. Next, we will present some of the pose estimators that could be combined with fiducial extraction.

12.2.3.1 Analytical Algorithm

Analytical methods use a reduced number of points. Their complexity and execution time are low. Several analytical algorithms have appeared in the literature [5,6], which essentially differ in the technique of resolution and the number of points they use. Didier [17] developed an analytical algorithm based on coded square targets. The method requires knowledge of:

- Intrinsic parameters of the camera
- Coordinates of the four corners of the target (named A, B, C and D) in the image
- The real size of a fiducial side

The algorithm is composed of two parts. The first part consists of computing the real depths of fiducial vertices and the second part is the pose computation.

The fiducial has a square shape, so it has the following property:

$$\overrightarrow{AB} = \overrightarrow{CD}. \tag{12.7}$$

Applying the perspective model of the camera, one gets the following expression:

$$\begin{pmatrix} u_B & -u_C & u_D \\ v_B & -v_C & v_D \\ -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} Z_B \\ Z_C \\ Z_D \end{pmatrix} = \begin{pmatrix} v_A \\ u_A \\ -1 \end{pmatrix}. \tag{12.8}$$

Solving (12.8), the depth of the four square corners is given by the following formulae:

$$\begin{aligned}
Z_B &= \frac{1}{\det M}\left[u_A\left(v_C - v_D\right) + v_A\left(u_D - u_C\right) + \left(u_C v_D - u_D v_C\right)\right] \\
Z_C &= \frac{1}{\det M}\left[u_A\left(v_B - v_D\right) + v_A\left(u_D - u_B\right) + \left(u_D v_B - u_B v_D\right)\right] \\
Z_D &= \frac{1}{\det M}\left[u_A\left(v_B - v_C\right) + v_A\left(u_C - u_B\right) + \left(u_B v_C - u_D v_B\right)\right] \\
\det M &= \left(u_C v_D - u_D v_C\right) + \left(u_D v_B - u_B v_D\right) + \left(u_B v_C - u_D v_B\right)
\end{aligned} \tag{12.9}$$

Once the real depth is known, one determines the translation and the orientation of the fiducial toward the camera. The translation is determined using the fiducial center computed from the coordinates of fiducial vertices, A, B, C and D. The rotation matrix is given by the following three vectors:

$$r_1 = \frac{\overrightarrow{AB} + \overrightarrow{DB}}{\left\|\overrightarrow{AB} + \overrightarrow{DB}\right\|}, \quad r_2 = \frac{\overrightarrow{AC} - \overrightarrow{DB}}{\left\|\overrightarrow{AC} - \overrightarrow{DB}\right\|}, \quad r_3 = r_1 \wedge r_2.$$

12.2.3.2 Hybrid Orthogonal Iteration Algorithm

In this method, the pose estimation is formulated as error metric minimization based on collinearity in object space. Using an object space collinearity error metric, an iterative algorithm is derived to compute orthogonal rotation matrices. Further information can be found in Lu et al. [4]. Such algorithms converge to a solution but, in some cases, they could be trapped into local minima. To avoid this and simultaneously reduce the number of algorithm iterations, we choose to initialize the algorithm with the result of the analytical method in Sect. 12.2.3.1 rather than the weak perspective computation that is usually employed for initialization.

12.2.3.3 Extended Kalman Filter Algorithm

In this third approach, we use an extended Kalman filter (EKF) to estimate position and orientation of the object with respect to the camera coordinate frame. The EKF is a set of mathematical equations that provides an efficient computational model to estimate the state of a process by minimizing the mean of a squared error [18]. The EKF is applied to nonlinear systems with Gaussian zero mean process and measurement noise. The evolution model and measurement process are given by the following equations:

$$\begin{cases} x_k = f\left(x_{k-1}, w_{k-1}\right) \\ z_k = h\left(x_{k-1}, n_{k-1}\right) \end{cases} \tag{12.10}$$

where x_k is the state vector, w_k the process noise, z_k is the measurement vector and n_k the measurement noise.

In the first step of the EKF, which is time update, the state vector and the error covariance matrix are predicted using initial estimates of $\hat{x}_k$ and P_k, respectively. Once this step is completed, the estimates become the input for the measurement update (correction) step. With the updated information, the time update step projects the state vector and the error covariance matrix to the next time step. By repeating these two steps recursively, we estimate the state vector $\hat{x}$ that represents the pose parameters.

As described previously, the time update projects the system state vector and its covariance matrix from the current step k into the next step $k+1$. The measurement model represents the relationship between the system state vector and the camera measurement inputs. First, we need to define the state vector for the EKF. Since our goal is to estimate the camera pose, we use the rotation angles and the translation components $(\phi, \psi, \theta, t_x, t_y, t_z)$ to represent the system state. The measurement input is provided by the camera. We have to estimate six variables of the state vector, while the total measurement input is a 8×1 vector:

$$z = \left(u_1 \; u_2 \; u_3 \; u_4 \; v_1 \; v_2 \; v_3 \; v_4\right)^t. \tag{12.11}$$

Applying the camera perspective model to the 3D points, we have the following equations:

$$u_i = \frac{M_1 \cdot \mathbf{p_i} + t_x}{M_3 \cdot \mathbf{p_i} + t_z} \qquad v_i = \frac{M_2 \cdot \mathbf{p_i} + t_y}{M_3 \cdot \mathbf{p_i} + t_z}, \tag{12.12}$$

in which $\mathbf{p_i} = (x_i, y_i, z_i)$ represents the 3-D point in the object reference frame and $M_i, \; i = 1, 2, 3$ are the components of the perspective projection matrix of the camera, given in (12.6).

Time Update

The time update produces estimates $\hat{x}$ of the state vector and of the error covariance matrix P. The equations of projection are given by:

$$\begin{cases} \hat{x}_{k+1}^- = \hat{x}_k \\ P_{k+1}^- = A_k P_k A_k^t + Q_k \end{cases} \tag{12.13}$$

where Q represents the covariance matrix of the process noise and A is the transition matrix arbitrarily chosen as $A = I_9$, with I_9 being the 9×9 identity matrix.

Measurement Update

The general equations for the measurement update step in the EKF are given by:

$$\begin{aligned} z_{k+1} &= h\left(\hat{x}_k\right) + n_k \\ K_k &= P_k^- H_k^t \left(H_k P_k^{-1} H_k^t + V_k \Gamma_k V_k^t\right) \\ P_k &= \left(I - K_k H_k\right) P_k^- \end{aligned} \tag{12.14}$$

The first of (12.14) is the measurement function, while the second and third compute the Kalman gain and update the error covariance, respectively.

The next step is to tune the EKF parameters. For us, function h will be related to (12.12) since each $\mathbf{p}_i$ is a coordinate in the local frame associated to the fiducial. Γ and V are both considered to be representations of errors of measurement of about 1 mm for the considered distance between the fiducial and camera. H is the Jacobian matrix consisting of the partial derivatives of function h with respect to elements of the internal state of the filter.

By executing the time update and measurement update recursively, we can estimate the rotation angles and the translation vector of the camera coordinate frame according to the workspace coordinate frame.

12.2.3.4 Hybrid Extended Kalman Filter Algorithm

This method is simply a combination of the analytic algorithm with the EKF algorithm. Indeed, as we have already stated, the difficulty with the EKF algorithm lies in guessing the parameters for the first time. Thus, we may use the analytical algorithm to initialize the pose values to accurately estimate the EKF states.

12.2.4 Experimental Results

In this section, we present experimental results and a detailed evaluation of different localization methods. A comparison between these methods is performed in order to determine their relative performance. We compared our hybrid EKF (H-EKF) method to the three other algorithms which are the analytical algorithm, the hybrid OI (H-OI) and the EKF. The comparison between these algorithms is carried out according to the following criterions:

- *Execution time*.
- *Reconstruction error* which measures the pixel-to-pixel difference between feature points on the detected target in the image and the 3-D target model projection using the computed pose parameters.
- *Generalization error* which consists of projecting the targets which were not used for pose computation on the image plan and measuring the variation in pixels between the projected points of the 3-D models and the corresponding targets detected in the image.
- *Real camera-target distance estimation* which measures the difference between the evaluation of the estimated distance by the pose algorithm and the real distance given by the robot.

The experimental tests were realized using the following hardware configuration:

- Pentium III 1.1 GHz
- Matrox Meteor II frame grabber
- Sony XC-555 camera

Table 12.1. Mean reconstruction error and execution time for each algorithm

	Analytical	H-OI	EKF	H-EKF
Execution time (μs)	20	240	6,022	1,910
Reconstruction error (pixels)	0.84	0.77	0.2	0,12

12.2.4.1 Reconstruction Error and Execution Time

In the first experiment, the camera is moved by hand around the target object, the four algorithms estimate the pose parameters and we evaluate the reconstruction error in the image. Table 12.1 summarizes the results obtained for each algorithm over 5,000 computed poses. The error is estimated by re-projecting the object model on the image. We then measure the deviation between real target corners and the projected corners.

As expected, the analytical method is the fastest algorithm. However, because no optimizations are performed, it is also the algorithm with the most serious reconstruction error. The EKF is the slowest algorithm, mainly because its initialization is not sufficiently close enough to the optimal solution which results in slow convergence. Depending on a compromise in speed/accuracy trade-off, one can choose the most appropriate algorithm on the basis of these results.

12.2.4.2 Generalization Error

To determine the generalization error, we use two targets (their side is 6 cm long) with different codes. One of the targets is used for pose estimation. Then, we re-project the model of the targets onto them and measure distances in pixels between corner projection and real corners of the second fiducial. This way, we can estimate the generalization error, as in Fig. 12.6. When compared to other algorithms, the H-EKF and the analytical method presents the best performance in terms of generalization error. The overall error behavior of these two algorithms is stable and does not present jitter in images. EKF appears sometimes as the weakest algorithm, as it can sometimes diverge if the initialization of the filter is not sufficiently close the true solution.

12.2.4.3 Real Camera-Target Distance Estimation

In order to evaluate camera-target distance errors of the various algorithms, we use a calibration robot-bench which moves in two directions X and Y, as in Fig. 12.7. The camera is mounted on the robot-bench, the target (18 cm in side length) is fixed on the other side of the bench. The robot displacement is sampled in 1,939 positions. For each one of them, our four pose estimation algorithms are applied to compute the distance between the optical center of the camera and the target. We have classified the obtained values into ten classes

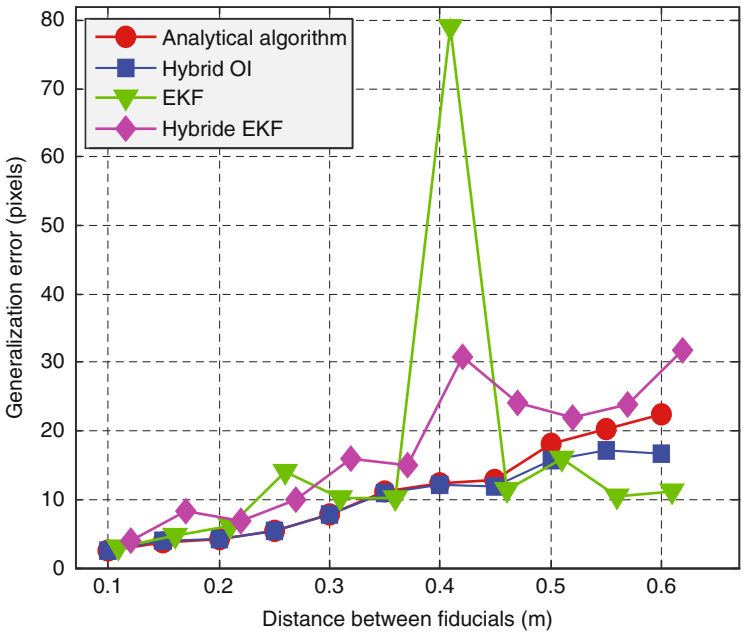

Fig. 12.6. Generalization error according to distance between fiducials

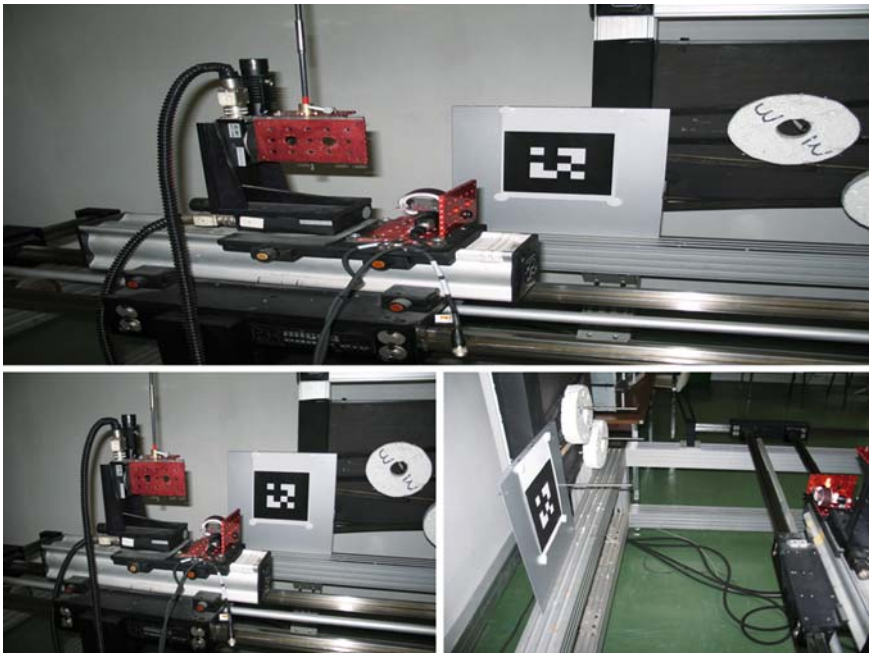

Fig. 12.7. Robot bench used for distance evaluation

(according to distance ranges between camera and target) and we computed the mean errors and variances of the pose estimation methods. The results are illustrated in Fig. 12.8, in which we compare the generated errors of the real distance given by the robot (robot position) and the position estimated by the pose algorithms. We notice that the analytical method presents a significant mean error relatively to other methods, however the error variance is quite small. The hybrid EKF and OI present the best performances. Finally, the EKF algorithm presents a large variance around its mean error.

Figure 12.9 shows real distances computed by the robot according to the distance estimated by the different pose algorithms. Indeed, this evaluation determines, with accuracy, the distance error generated from each pose estimator. The interpretation of errors is performed by approximating the curves in Fig. 12.9 with non linear regression for the hybrid OI, EKF and hybrid EKF algorithm and a quadratic regression for the analytical algorithm. The mean error of the OI algorithm is 0.81%, that is, a mean error of 8.1 mm for a distance of 1 m.

The hybrid OI error is estimated to 0.84%, while the EKF degenerates and presents a mean error of 2.6%. The lowest value of error is obtained with the hybrid EKF where it is estimated as 0.72%. We conclude that the hybrid EKF is best real distance estimator.

12.2.4.4 Virtual Object Rendering Results

Since the pose parameters were determined, we have projected a virtual cube on the detected real target in order to evaluate visually the virtual object rendering stability. In this experiment, the camera is freely moved around fiducials. The identification algorithm detects and tracks targets in frames and the hybrid EKF estimates position and orientation of the camera. In Fig. 12.10, we see that virtual objects are well superimposed on the real image and remain laid on the target for different camera poses.

12.2.4.5 Discussion

In this study, we compared the performances of four pose estimation algorithms. We evaluated these methods using an experimental protocol to compute several error sources and estimate real distances. We used three iterative methods based on nonlinear optimization and a new analytical method based on direct computation of parameters.

Indeed, the two kinds of algorithms have both advantages and shortcomings. Iterative methods are accurate, but suffer from high computation expense due to inaccurate initialization and local minima problems. On the other side, the analytical methods are fast, but their major disadvantage is their lack of accuracy.

Table 12.2 summarizes the results obtained from the different pose estimation algorithms for each experiment criterion. Clearly, purely analytical

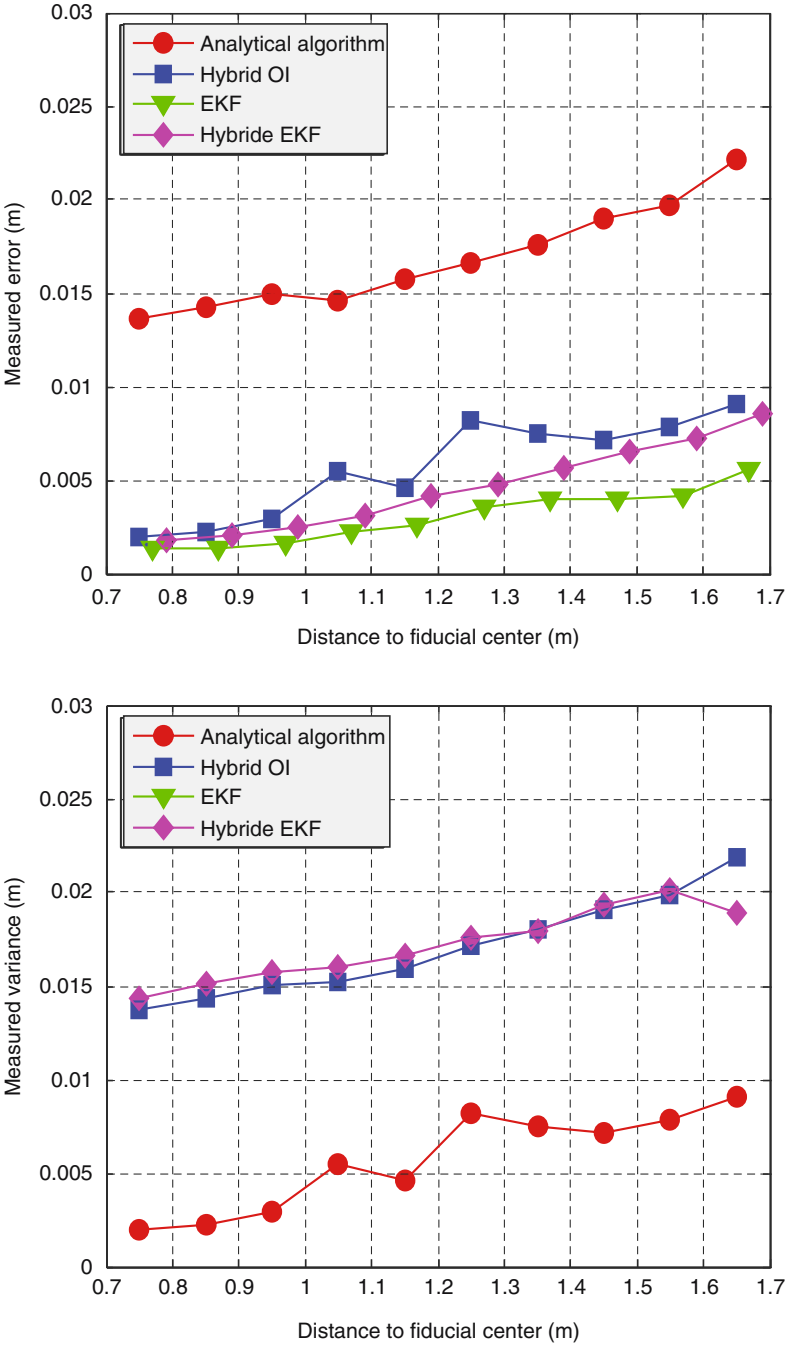

Fig. 12.8. Mean errors and variances of the classified data

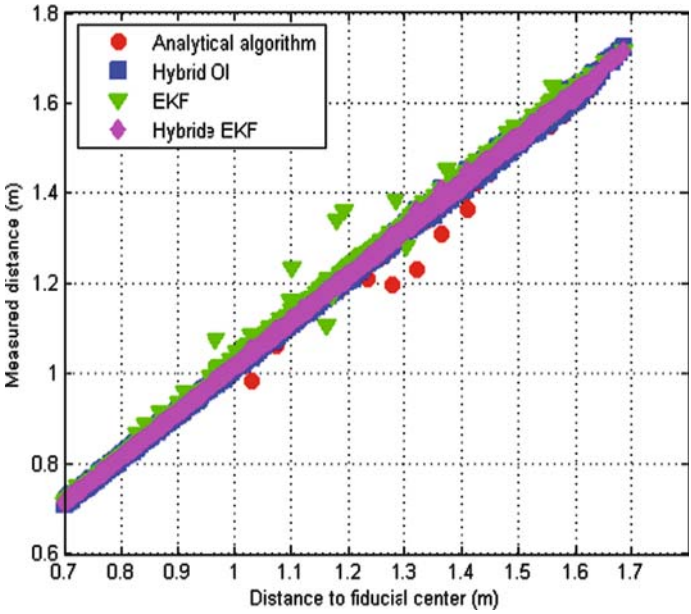

Fig. 12.9. Evaluation of measured distances according to real distances

methods and purely iterative methods rank badly whereas hybrid methods seem to be the best compromise for an algorithm that attempts to satisfy all the criteria we set. However, depending on the individual criteria an algorithm must fulfill, one of the presented methods may be more appropriate.

Target tracking is widely used in AR applications. Such an interest could be explained by the intrinsic strength of the previous algorithms:

- Their requirements, in terms of image processing, are quite low since their geometric properties are chosen so that they can be easily detected.
- They use their own code which makes it easy to distinguish one from another. Moreover this code can be semantically linked to the application.

These systems also have flaws:

- Target systems invade the scene they need to be placed inside the environment or stuck on the objects we wish to track.
- They cannot be applied to every type of environment. For example, in some ARS the target may be soiled in industrial facilities, hence reducing the tracking efficiency.
- They cannot be placed in arbitrary locations in the scene; usually targets must be placed on objects or places with planar surfaces.

These limitations of marker-based tracking have led to research and development of so-called *marker-less tracking* which does not require additional scene instrumentation but is more involved in terms of image processing and

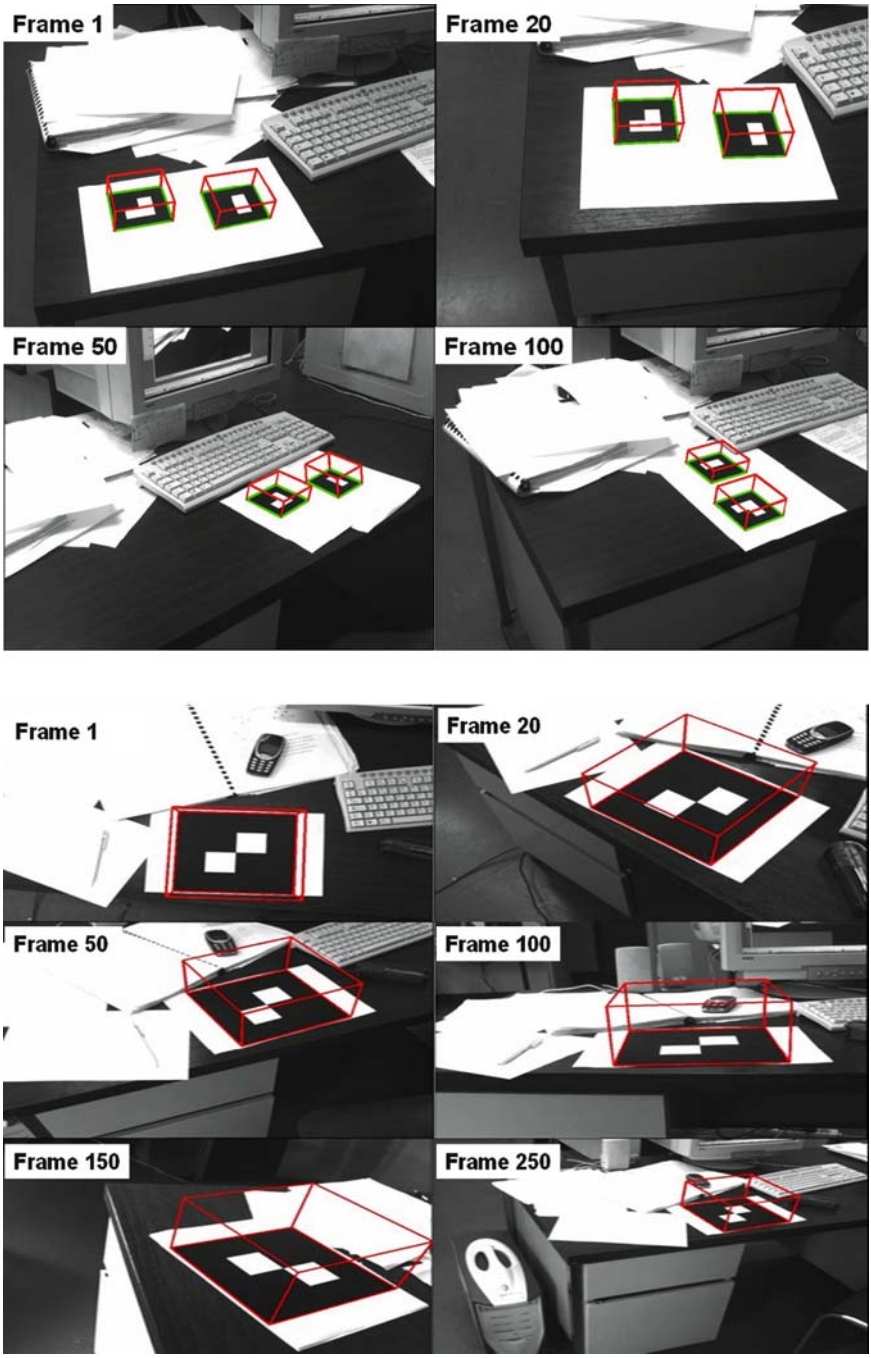

Fig. 12.10. Virtual object overlay in a tracking sequence using various fiducials

Table 12.2. Overall results of the different algorithms

	Analytical	H-OI	EKF	H-EKF
Execution time	First	Second	Fourth	Third
Reconstruction error	Fourth	Third	Second	First
Generalization error	Second	First	Third	Third
Distance estimation	Fourth	Third	First	Second
Mean ranking	2.75	2.25	2.5	2.25

computations. The interest in these methods has been renewed over the past few years, following the rapid increase of available inexpensive computing power, which makes possible the real time execution of marker-less algorithms to track complex objects.

12.3 Marker-Less Based Approaches

Marker-less tracking is a very complex task, as it uses image processing operators to detect natural features in the video stream and to recover camera position and orientation. Several marker-less tracking approaches have been developed in the recent years. Model-based tracking approaches appear to be the most promising among the standard vision techniques currently applied in AR applications. The main idea of the model-based techniques is to identify features in the images using an object model. The problem is solved using registration techniques that allow alignment of 2-D image data and a 3-D model. Edge features are widely used to track an object in image sequences. Wuest et al. [19] present a model-based line tracking approach that can handle partial occlusion and illumination changes. To obtain robust 2-D-to-3-D correspondences, they have implemented a multiple hypotheses assigning method using the Tukey estimator. The camera pose is computed by minimizing the distances between the projection of the model lines and the most likely matches found in the image.

Drummond and Cipolla [20] propose a novel framework for 3-D model-based tracking. Objects are tracked by comparing projected model edges to edges detected in the current image. Their tracking system predicts the edge locations in order to rapidly perform the edge search. They have used a Lie group formalism in order to transform the motion problem into simple geometric terms. Thus, tracking becomes a simple optimization problem solved by means of iterative reweighed least squares.

Yoon et al. [21] present a model-based object tracking to compute the camera 3-D pose. Their algorithm uses an Extended Kalman Filter (EKF) to provide an incremental pose-update scheme in a prediction-verification framework. In order to enhance the accuracy and the robustness of the tracking

against occlusion, they take into account the measurement uncertainties associated with the location of the extracted image straight-lines.

Recently, Comport et al. [22] have proposed a real-time 3-D model-based tracking algorithm. They have used a visual control approach to formulate the pose estimation problem. A local moving edges tracker is implemented which is based on tracking of points normal to the object contours. In order to make their algorithm robust, they have integrated a M-estimator into the visual control law.

Other approaches have also been applied where different features have been combined to compute the camera pose, such as edge and point feature combination [23] and edge and texture information combination [24, 25]. In the next section, we will present a robust line tracking approach for camera pose estimation which is based on particle filtering framework [26]. This will illustrate to the reader how to conceive and to evaluate such a system.

12.3.1 Marker-Less Line Tracking Approach

12.3.1.1 Problem Definition

In this section, we set the 3-D constraints for pose determination when using line features. Given correspondences between 3-D and 2-D lines found in the image, the goal is to find the rotation matrix and the translation vector which map the world coordinate system to the camera coordinate system. Let L be an object line. Several representations for a 3-D line have been proposed [27]. In our approach, we represent the 3-D line L by its two end-points $\mathbf{p}_1$ and $\mathbf{p}_2$ (see Fig. 12.11). The point $\mathbf{p}_i$ in world coordinates can be expressed in camera frame coordinates as in (12.1).

Let $(x_1,\ y_1,\ z_1)$ and $(x_2,\ y_2,\ z_2)$ be the camera coordinates of the end-points $\mathbf{p}_1$ and $\mathbf{p}_2$ which project onto the image plane at $\mathbf{m}_1$ and $\mathbf{m}_2$ respectively. The projection plane formed by the image line $(\mathbf{m}_1\mathbf{m}_2)$ is given by the plane $(O\mathbf{m}_1\mathbf{m}_2)$. The 3-D line L_1 must lie in this plane. The normal $\vec{N}$ to the projection plane is given by:

$$\vec{N} = \vec{n}_1 \times \vec{n}_2, \tag{12.15}$$

where $\vec{n}_1$ and $\vec{n}_2$ are the optical rays of the image points $\mathbf{m}_1$ and $\mathbf{m}_2$. Thus, the 3-D line constraint can be formulated as:

$$\vec{N} \cdot (R\mathbf{p}_i + T) = 0. \tag{12.16}$$

The 3-D line constraint represents the fact that any point on the 3-D line in camera coordinates ideally must lie in the projection plane. This constraint relates both rotation and translation pose parameters to the 3-D model and 2-D image lines. In the next section, we will describe the use of this constraint within a particle filter to estimate the camera pose.

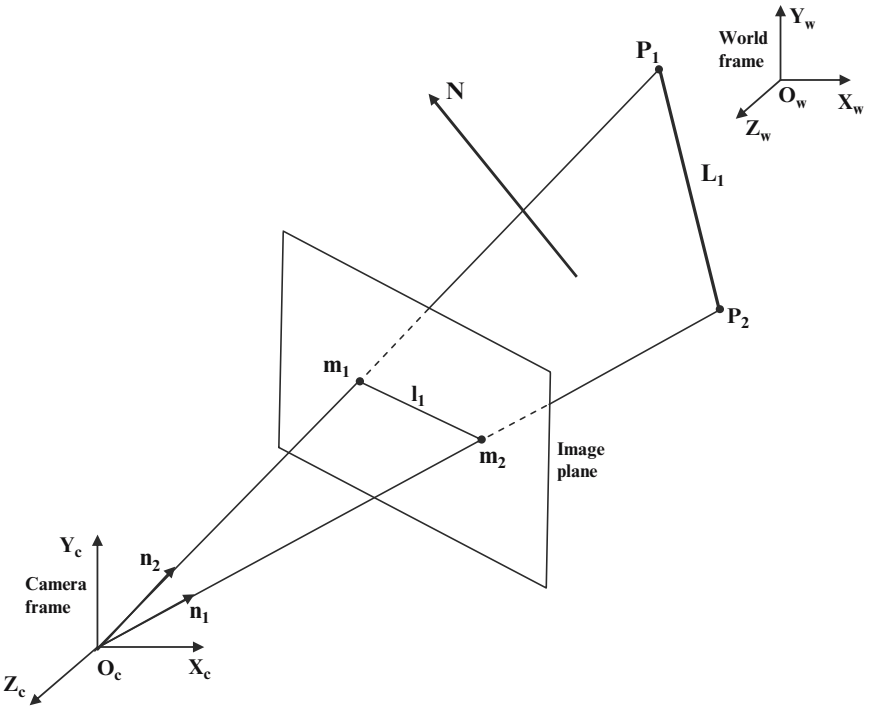

Fig. 12.11. Perspective projection of 3D line

12.3.1.2 Particle Filter Implementation

The particle filter is used to estimate the posterior density for the 3-D camera pose parameters. The camera state is represented by position and rotation of the camera with respect to a world coordinate system. Rotations can be represented with various mathematical entities, such as matrices, axes and angles, Euler angles, and quaternions. However, quaternions have proven very useful in representing rotations because of several advantages over other representations, such as increased compactness, lesser susceptibility to round-off errors, avoidance of discontinuous jumps.

A quaternion representation of rotation R is written as a normalized four dimensional vector $q = \lfloor q_0 \; q_x \; q_y \; q_z \rfloor$, where $\left(q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1\right)$.

Thus, the camera state is given by:

$$X = \lfloor q_0 \; q_x \; q_y \; q_z \; t_x \; t_y \; t_z \rfloor, \tag{12.17}$$

where $T = [t_x \; t_y \; t_z]^T$ is the camera position (translation) vector.

We denote the camera state at time k by the vector X_k. Each particle X_k^n corresponds to a potential pose of the camera. The most probable particle will have important weights. These provide an approximation to the posterior density. Basically, the key components of the Particle Filter are the state

dynamics and the observations used. More details on particle filter theory are given in [28, 29].

State Dynamics

The particle filter requires a probabilistic model for the state evolution between time steps, i.e. the density $p\left(X_x \left| X_{k-1}\right.\right)$. Since we have no prior knowledge of camera motion, we use a simple random walk based on a uniform density about the previous camera state [30]:

$$p\left(X_k \left| X_{k-1}\right.\right) = U\left(X_{k-1} - v, X_{k-1} + v\right), \tag{12.18}$$

where $v = \left[v_1 \; v_2\right]^T$ represents the uncertainty about the incremental camera movement (v_1 for rotation and v_2 for translation).

As the camera undergoes a random walk, it moves a certain random distance Δd and deviates from its previous direction by some random quantity $\Delta\theta$. The proposed *Uniform Random Walk* model has a probability density distributed according to $v_i \cdot (2 \cdot Rand - 1) \;\; i = 1, \; 2$, with the random variable *Rand* uniformly distributed between 0 and 1. The parameters v_i are set empirically.

Observation Model

Let y_k be the observation at frame k and $y_{1:k}$ the set of observations from frame 1 to frame k. In our case, observations correspond to the extracted image lines l_i (see Fig. 12.1). We also assume that we have a set of 3-D scene lines, $Z = \{L_1, \, L_2, \, \ldots, \, L_M\}$, which are defined in the world coordinate frame. Each line L_i is represented by its two end-points P_1^i and P_2^i, respectively. The projection of the line L_i on the camera frame with the camera state X_k is then denoted $C(L_i, X_k)$ and is given by:

$$C\left(L_i, X_k\right) = \begin{cases} R_k \cdot \mathbf{p_1^i} + T_k \\ R_k \cdot \mathbf{p_2^i} + T_k \end{cases} \tag{12.19}$$

Equation (12.19) describes the projection of the 3-D model lines in the camera coordinates frame. Thus, the two end points $\mathbf{p_1^i}$ and $\mathbf{p_2^i}$ of the 3-D line L_i are projected in the camera frame using the rigid transformation between the world coordinates frame and the camera coordinates frame, given by the current camera pose parameters (R_k, T_k).

The solution of the particle filter is to obtain successive approximations to the posterior density $p\left(X_k \left| y_{1:k}, Z\right.\right)$. This is generally provided in the form of weighted particles $\left\{\left(X_k^1, w_k^1\right), \ldots, \left(X_k^n, w_k^n\right)\right\}$, where X_k^n is a state space sample and the weights w_k^n are proportional to $p\left(y_k \left| X_k^n\right.\right)$, so that:

$$\sum_{n=1}^{S} w_k^n = 1 \tag{12.20}$$

for a total of S particles.

The likelihood $p\left(y_k \left| X_k, Z\right.\right)$ is based on the closeness of the projected line $C\left(L_i,\ X_k\right)$ to get on the projected plane defined by the vector $\vec{N}$. In other words, we propose to use the 3-D line constraint in (12.16) to construct the likelihood $p\left(y_k \left| X_k, Z\right.\right)$. Practically, for each extracted image line $l_i (i = 1, \ldots, l)$ we compute the normal vector $\vec{N}_i$ to its projection plane. Then, for each particle X_k^n we determine all the projected lines corresponding to the model 3-D lines L_j as follows:

$$C\left(L_j, X_k^n\right) \qquad n = 1, \ldots, S \text{ and } j = 1, \ldots, M \tag{12.21}$$

To compute the likelihood we use a function related to the number of the model 3-D lines whose projections into the camera frame are within a given threshold of extracted projection planes, i.e.

$$p\left(y_k \left| X_k, Z\right.\right) = \exp\left\{-\sum_{i=1}^{l}\sum_{j=1}^{M} d_l\left(\vec{N}_i, L_j, X_k\right)\right\}, \tag{12.22}$$

where I and M corresponds here to the number of 2-D extracted image lines and the number of the 3-D model lines, respectively. In (12.22), $d_l\left(\vec{N}_i, L_j, X_k\right)$ indicates whether the 3-D line L_j is an inlier or outlier with respect to the observation $\vec{N}_i$ and the state X_k, i.e.

$$d_l\left(\vec{N}_i, L_j, X_k\right) = \begin{cases} 1 & if\ \vec{N}_i.\vec{C}\left(L_j, X_k\right) < \varepsilon_l \\ 0 & otherwise, \end{cases} \tag{12.23}$$

where ε_l is a threshold which defines the minimal distance to the plane projection.

Equation (12.23) implies that if the projection of the 3-D model line in the camera coordinate frame is orthogonal to the normal $\vec{N}_i$ to the projection plane, then we consider the 2-D-to-3-D matching between the image line l_i and the 3-D line L_j as correct and allot to it a score equal to 1. Otherwise, we consider the 2-D-to-3-D matching line as false and we put the score to zero. Indeed, any point on the 3-D model line, in camera coordinates frame, must lie in the projection plane, which justifies our modeling strategy.

Finally, the weights w_k^n, for both point and line features, are given by:

$$w_k^n = \frac{p\left(y_k \left| X_k^n, Z\right.\right)}{\sum_{n}^{S} p\left(y_k \left| X_k^n, Z\right.\right)} \tag{12.24}$$

The output of the particle filter is given by:

$$\hat{X}_k = \sum_{n=1}^{S} w_k^n \cdot X_k^n \tag{12.25}$$

To avoid the degeneracy of the particle filter method, a re-sampling stage may be used to eliminate samples with low importance weights and multiply samples with high importance weights (see Sect. 12.2). In our work, we have implemented the selection scheme proposed by Gordon [28].

12.3.1.3 Experimental Results

In order to study the robustness of our algorithm, we have used a complex test sequence containing a 3-D object, as in Fig. 12.12, and simulated the camera pose tracking in uncontrolled environment. This sequence test enables us to evaluate our method in more realistic circumstances. This sequence is recorded from a moving camera pointing toward the object of interest. The frame rate is 25 frames/s (25 Hz) with a sequence duration of 40 s. The resolution of the collected images is 320×240 pixels.

Frame 1 is used to calibrate the camera and also to initialize the camera tracking algorithm. Thus, at $k = 1$ the state vector X_1 is initialized with the camera pose parameters given by the calibration procedure. In addition, to extract line features from the current image, we have used the well known Hough line transform. Each 2D line is then defined by its two end-points in the image plane. This process generates more lines than needed to determine a model pose, thus only a small subset of them are used by the algorithm to compute pose and correspondence.

In order to analyze the pose estimation accuracy, we define the image registration error (in pixels) which corresponds to the distance between the

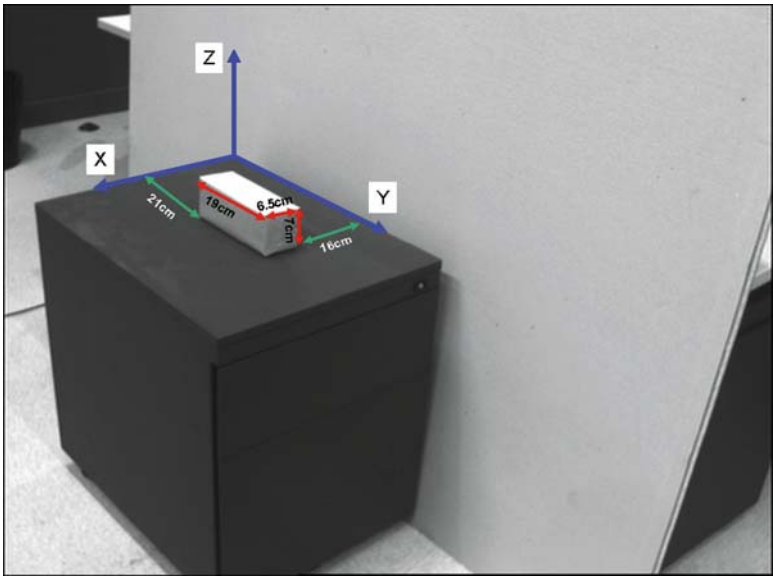

Fig. 12.12. 3D object used in experiments

detected features in the image (inliers) and the re-projected 3-D reference features using the estimated camera pose. When the camera pose is estimated, we re-project the 3-D model on the current frame. This gives a visual tool to assess the tracking accuracy. If augmented graphics appear stable in their correct position as the camera moves, this indicates good tracking performance.

Figure 12.13a demonstrates successful tracking. Clearly, the virtual object is well superimposed on the real world. An analysis of the results shows that our algorithm performs quite accurately for AR applications. Indeed, the system exhibits an average image error lower than 1%.

In addition to the previous, we have developed an implementation to test the robustness of our approach to partial occlusion of the object. Figure 12.13b shows that the virtual object is correctly augmented onto the image frame although the real object is occluded at approximately 40%. This demonstrates that our algorithm can accurately estimate the camera pose when severe occlusion occurs. Indeed, this robustness is simply ensured by the observation model of the particle filter, which uses inliers with respect to the observations in order to compute the filter output. Furthermore, as the 3-D constraint equation for the camera pose parameters was developed in the case of "infinite image line," any image points on the 2-D line can be used to construct the corresponding projection plane. Thus, when partial occlusion occurs, it is sufficient to detect only small parts of the fiducial edges to estimate the camera pose.

Furthermore, it is essential to minimize the number of particles used in the estimation step. The number of particles needed is determined so that the computational load remain sufficiently low for real-time application, while at the same time high performance is assured. We have performed several experiments to determine the appropriate number of particles and found that $N = 200$ leads to a good compromise.

Finally, real-time performance of our tracking approach has been achieved by carefully evaluating the processing time to compute the filter output. We have implemented our algorithm on an Intel Pentium IV 2.99 GHz PC equipped with a standard Matrox Meteor II acquisition card and an iS2 IS-800 CCD camera. The computational time depends mainly on the number of extracted lines. As our goal is to demonstrate the feasibility of the proposed tracking framework, we have tuned the threshold parameters of the Line Hough Transform to produce only interesting lines, that is long straight lines. As a result, the processing time of our algorithm is, in average, less than 20 ms.

12.3.1.4 Discussion

We have presented an implementation example of a camera pose estimation algorithm based on lines tracking. First, we formulated the problem by highlighting the geometrical constraint which relate the pose parameters to the 3-D model and 2-D image lines using the computer vision paradigm. Then, we

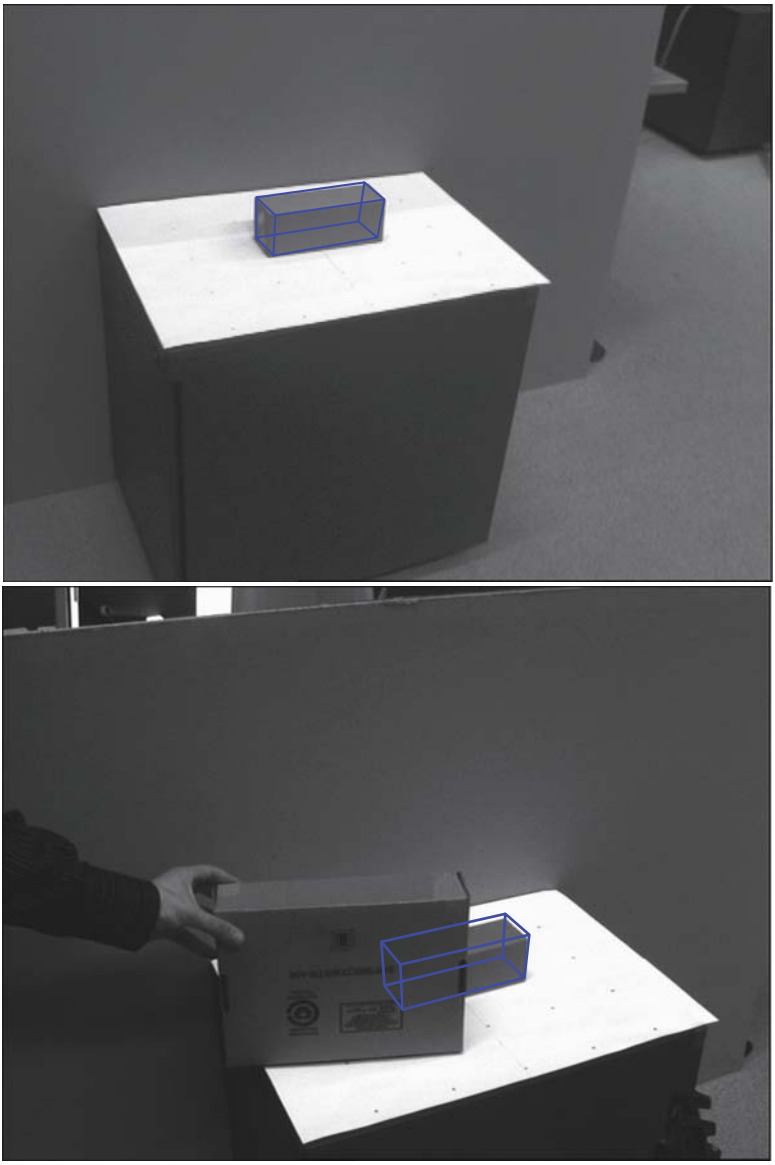

Fig. 12.13. Experimental results. (**a**) Example of camera pose estimation result (**b**) camera tracking under partial occlusion of the 3D object

demonstrated the way to use this constraint in a particle filter to estimate the camera pose. To evaluate our algorithm performance, we defined several criteria which take into account AR applications needs, such as registration error, robustness against occlusion and computational load. Experimental results show that our algorithm can track the camera pose successfully and accurately

under various conditions, including severe occlusion. The achieved performance is good compared to the performance of other line tracking techniques.

However, we find that the performance, in terms of speed, accuracy and flexibility, of marker-less tracking techniques in general is still beyond what real–world AR applications demand. Hybrid approaches can be an interesting tracking solution, where other sensors (e.g. inertial sensors) are used to compensate vision-based tracking. Indeed, the fusion of complementary sensors is used to build better tracking systems. Synergies can be exploited to gain robustness, tracking speed and accuracy, and to reduce jitter and noise.

12.4 Hybrid Approaches

Hybrid solutions attempt to overcome the drawbacks of any single sensing solution by combining the measurements of at least two tracking methods. Nowadays, a hybrid tracking system seems to be the best solution to achieve a better vision-based camera tracking, and is widely applied in recent ARS. State et al. [31] developed a hybrid tracking scheme which combined a fiducial–based vision tracker with a magnetic tracker. Their system exhibits the (static) registration accuracy of vision-based trackers and the robustness of magnetic trackers.

Auer and Pinz [32] created a similar magnetic–vision system, which employs the corners as visual features. In their solution, prediction from the magnetic tracker is used to reduce the search areas in the optical tracking subsystem achieving a faster and more robust tracking. Another popular choice in unprepared environments is inertial and natural feature video–based tracker fusion. You et al. [33] created a tracking system which combined a natural feature vision system with three gyro sensors. The fusion approach is generally based on the structure from motion (SFM) algorithm, in which approximate feature motion is derived from inertial data, and vision feature tracking corrects and refines these estimates in the image domain.

Chen and Pinz [34] presented a structure and motion framework for real time tracking combining inertial sensors with a vision system based on natural features. Their model uses fusion data to predict the user's pose and also to estimate a sparse model of the scene without any visual markers. An Extended Kalman Filter (EKF) is used to estimate motion by fusion of inertial and vision data and a bank of separate filters to estimate the 3-D structure of the scene.

Chai et al. [35] employs an adaptive pose estimator with vision and inertial sensors for overcoming the problems of inertial sensor drift and vision sensor slow measurement. The EKF is also used for data fusion and error compensation. Foxlin and Naimark [36] have developed the VIS-tracker system, which fuses data from inertial and vision sensors. They use a novel 2-D barcode system to recognize a large number of unique fiducial codes so as to initialize its location over a wide area. Their system is robust to lighting conditions variation, fast motions, occlusions, and has very low latency.

Recently, Ababsa and Mallem [37] proposed a real-time hybrid approach for 3-D camera pose estimation that integrates inertial and vision-based technologies. A fast and accurate vision based corner tracker forms the basis of their vision system. In order to fuse sensor data, they propose to use a Particle Filter instead of the EKF.

Furthermore, mobile outdoor ARS use, in addition to the camera, a GPS for position measurements and inertial sensors coupled with magnetic compasses for orientation. Examples of such systems include the prototype of the university of Columbia [38], the Tinmith-metro system [39], the ARVino System [40], and the Going out System developed by Reitmayr and Drummond [41].

Hybrid tracking is still a great challenge. Current systems require an extensive calibration of all the sensors. In some cases, the system has to be initialised after a few minutes because of the drift. Moreover, most of the vision-based tracking techniques mentioned here assume a known and accurate initial position.

With the previous in mind, future research should be focused on robust and reliable algorithms for hybrid marker-less tracking, initialization and sensor fusion problems. Thus, future ARS are expected to be able to automatically select and combine the suitable algorithms for defined conditions, to fuse, filter and estimate the camera pose.

References

1. D.G. Lowe. Fitting Parameterized Three-Dimensional Models to Images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13, pp. 441–450, 1991.
2. R.M. Haralick. Pose Estimation from Corresponding Point Data. IEEE Transactions on Systems, Man, and Cybernetics, 19(6), pp. 1426–1446, 1989.
3. D.F. DeMenthon and L.S. Davis. Model-based Object Pose in 25 Lines of Code. International Journal of Computer Vision, 15(1–2), pp. 123–141, 1995.
4. C.P. Lu, G. Hager, and E. Mjolsness. Fast and Globally Convergent Pose Estimation from Video Images. IEEE Transaction on Pattern Analysis and Machine Intelligence, 22(6), pp. 610–622, June 2000.
5. Y. Hung, P. Yeh, and D. Harwood. Passive ranging to known planar point sets. In Proceedings of IEEE International Conference on Robotics and Automation, 1, pp. 80–85, St. Louis, Missouri, 1985.
6. L. Quan and Z. Lan. Linear n-Point Camera Pose Determination. IEEE Transactions. Pattern Analysis and Machine Intelligence, 21(7), pp. 774–780, July 1999.
7. Y. Cho and U. Neumann. Multi-Ring Color Fiducial Systems for Scalable Fiducial Tracking Augmented Reality. In Proceedings of the Virtual Reality Annual International Symposium (VRAIS'98). pp. 212, Washington, DC, USA, 1998.
8. L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In Proceedings of ACM/IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2002). pp. 27–36, Darmstadt, Germany, 2002.

9. J. Rekimoto. Matrix: A Realtime Object Identification and Registration Method for Augmented Reality. In Proceedings of the Third Asian Pacific Computer and Human Interaction (APCHI'98). pp. 63–68, Washington DC, USA, 1998.
10. J. Rekimoto and Y. Ayatsuka. Cybercode: Designing Augmented Reality Environments with Visual Tags. In Proceedings of DARE 2000 on Designing Augmented Reality Environments, pp. 1–10, Elsinore, Denmark, 2000.
11. H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In Proceedings of the 2nd ACM/IEEE International Workshop on Augmented Reality (IWAR'99), pp. 85–92, Washington DC, USA, 1999.
12. X. Zhang, S. Fronz, and N. Navab. Visual Marker Detection and Decoding in AR Systems: A Comparative Study. In Proceedings of the ACM/IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2002), pp. 97, Washington, DC, USA, 2002.
13. C.B. Owen, X. Fan, and P. Middlin. What is the Best Fiducial? In Augmented Reality Toolkit, The First IEEE International Workshop. IEEE, 2002.
14. M. Fiala. Artag, A Fiducial Marker System Using Digital Techniques. In Proceedings of the 2005 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'05). 2, pp. 590–596, Washington, DC, USA, 2005.
15. J. Canny. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), pp. 679–698, 1986.
16. Z. Zhang. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In Proceedings of the International Conference on Computer Vision, 1, pp. 666, Corfu, Greece, 1999.
17. J.Y. Didier. Contributions à la déxtérité d'un système de réalité augmentée mobile appliquée à la maintenance industrielle. In PhD Thesis, Université d'Évry, France, 2005.
18. G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report No. TR 95-041, Department of Computer Science, University of North Carolina, USA, 2004.
19. H. Wuest, F. Vial, and D. Stricker. Adaptive Line Tracking with Multiple Hypotheses for Augmented Reality. In Proceedings of ACM/IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2005), pp. 62–69, Vienna, Austria, October 2005.
20. T. Drummond and R. Cipolla. Real-Time Visual Tracking of Complex Structures, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7), pp. 932–946, July 2002.
21. Y. Yoon, A. Kosaka, J.B. Park, and A.C. Kak. A New Approach to the Use of Edge Extremities for Model-based Object Tracking. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005). pp. 1883–1889, Barcelonna, Spain, April 2005.
22. A.I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework. IEEE Transactions on Visualization and Computer Graphics, 12(6), 615–628, July/August 2006.
23. V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua. Fully Automated and Stable Registration for Augmented Reality Applications. In Proceedings of ACM/IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2003), p. 93, Tokyo, Japan, 2003.

24. L. Vacchetti, V. Lepetit, and P. Fua. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In Proceedings of ACM/IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2004), pp. 48–57, Arlington, VA, November 2004.
25. M. Pressigout and E. Marchand. Real-Time 3D Model-based Tracking: Combining Edge and Texture Information. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 06), pp. 2726–2731, Orlando, Florida, May 2006.
26. F. Ababsa and M. Mallem. Robust Line Tracking Using a Particle Filter for Camera Pose Estimation. In Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST 2006), pp. 207–211, Limassol, Cyprus, November 2006.
27. A. Beutelspacher and U. Rosenbaum. Projective Geometry: From Foundations to Applications, Cambridge University Press, Cambridge, 1998.
28. N.J. Gordon. A Hybrid Bootstrap Filter for Target Tracking in Clutter. IEEE Transactions on Aerospace and Electronic Systems, 33, pp. 353–358, 1997.
29. A. Doucet, N. de Freitas, and N. Gordon. Sequential Monte Carlo Methods in Practice. Springer, Berlin Heidelberg New York, 2001.
30. M. Pupilli and A. Calway. Real-Time Camera Tracking Using a Particle Filter. In Proceedings of the British Machine Vision Conference (BMVC 2005), pp. 519–528, Oxford, UK, September 2005.
31. A. State, G. Hirota, D.T. Chen, W.F. Garrett, and M.A. Livingston, Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96), pp. 429–438, New York, NY, USA, 1996.
32. T. Auer and A. Pinz. Building a Hybrid Tracking System: Integration of Optical and Magnetic Tracking. In Proceedings of the 2nd ACM/IEEE International Workshop on Augmented Reality (IWAR'99), pp. 13–19, Washington, DC, USA, 1999.
33. S. You, U. Neumann, and R. Azuma. Hybrid Inertial and Vision Tracking for Augmented Reality Registration. In Proceedings of IEEE International Conference on Virtual Reality (VR 99), pp. 260–267, 1999.
34. J. Chen and A. Pinz. Structure and Motion by Fusion of Inertial and Vision-based Tracking. In OCG. Proceedings of the 28th OAGM/AAPR Conference. Digital Imaging in Media and Education (W. Burger and J. Scharinger, eds.), pp. 55–62, 2004.
35. L. Chai, W. Hoff, and T. Vincent. Three-Dimensional Motion and Structure Estimation Using Inertial Sensors and Computer Vision for Augmented Reality. Presence: Teleoperators and Virtual Environments, pp. 474–492, 2002.
36. E. Foxlin and L. Naimark. VIS-Tracker: A Wearable Vision-Inertial Self-Tracker. In Proceedings of the IEEE Conference on Virtual Reality (VR 2003), pp. 193, Los Angeles, CA, USA. March 2003.
37. F. Ababsa and M. Mallem. Hybrid 3D Camera Pose Estimation Using Particle Filter Sensor Fusion. Advanced Robotics. International Journal of the Robotics Society of Japan (RSJ), pp. 21, 165–181, 2007.
38. S. Feiner, B. MacIntyre, T. Hollerer, and A. Webster. A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. In Proceedings of the 1st IEEE International Symposium on Wearable Computers (ISWC 97). pp. 74, Washington, DC, USA, 1997.

39. W. Piekarski and B. Thomas. Tinmith-metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer. In Proceedings of the 5th International Symposium on Wearable Computers, pp. 31–38, Zurich, 2001.
40. G.R. King, W. Piekarski, and B.H. Thomas. ARVino – Outdoor Augmented Reality Visualisation of Viticulture GIS Data. In Proceedings of the ACM/IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2005), pp. 52–55, Washington, DC, USA, 2005.
41. G. Reitmayr and T. Drummond. Going out: Robust Model-based Tracking for Outdoor Augmented Reality. In Proceedings of the ACM/IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2006), pp. 109–118, Santa Barbara, CA, USA, October 2006.

A. Websites and Companies Specializing in Augmented Reality Research and Applications

http://www.isense.com/
http://www.xsens.com/
http://www.augmented-reality.org/
http://www.igd.fhg.de/index.html.en
http://studierstube.icg.tu-graz.ac.at/
http://www.miralab.unige.ch/
http://vrlab.epfl.ch/
http://www1.cs.columbia.edu/cvgc/
http://www.tinmith.net/
http://www.arvika.de/
http://ar.in.tum.de/Chair/ProjectDwarf
http://evra.ibisc.univ-evry.fr/index.php/AMRA
http://evra.ibisc.univ-evry.fr/index.php/ARCS
http://www.ibisc.univ-evry.fr/Equipes/RATC/
http://www.hitl.washington.edu/artoolkit/