Journal of Electronic Imaging

SPIEDigitalLibrary.org/jei

# Real-time camera tracking using a particle filter combined with unscented Kalman filters

Seok-Han Lee

Journal of Electronic Imaging 23(1), 013029 (Jan–Feb 2014)

# Real-time camera tracking using a particle filter combined with unscented Kalman filters

**Seok-Han Lee***
Jeonju University, Department of Information and Communication Engineering, 303, Chunjam-Ro, Wansan-Gu, Jeonju, Jeollabuk-Do, 156-759, Korea

**Abstract.** Real-time camera tracking is steadily gaining importance due to the drive from various applications, such as augmented reality, three-dimensional structure estimation/modeling, and mobile computing environment. However, tracking a monocular camera in an unknown environment is not a trivial work. We describe a real-time camera tracking framework designed to track a monocular camera in a workspace. In particular, we focus on integration of a bundle of nonlinear filters to achieve robust camera tracking and scalable feature mapping, which can extend to larger environment. The basic idea of the proposed framework is that a particle filter–based camera tracking is connected to independent feature tracking filters, which have fixed-state dimension. In addition, every estimate required for template prediction is obtained from the independent feature estimators so that the template prediction can be maintained without additional framework for the template state estimation. We split the camera tracking and feature mapping into two separate tasks, and they are handled in two parallel processes. We demonstrate the effectiveness of the proposed approach within a desktop environment in real time.  [DOI: 10.1117/1.JEI.23.1.013029]

Keywords: camera tracking; augmented reality; visual tracking.

Paper 13489 received Sep. 6, 2013; revised manuscript received Dec. 24, 2013; accepted for publication Jan. 28, 2014; published online Feb. 26, 2014.

## 1 Introduction

Typically, vision-based augmented reality (AR) systems operate on the basis of prior knowledge of the environment. This could be a three-dimensional (3-D) model or a fiducial marker, which is known to be present in the scene. The application then allows a user to interact with this environment based on the prior information on this model. If the known model or fiducial is accurate, registration can be performed directly from it, and this is the conventional method of vision-based AR system. Quite often, however, accurate information is not available from the scene, and this limits range and accuracy of registration. Therefore, there have been considerable research efforts for the techniques known as real-time camera tracking, in which the system attempts to add unknown features to its feature map, and these then provide registration even when the original reference map is out of visible range. Typically, vision-based tracking systems operate on the basis of a monocular camera. One problem of the single camera-based systems is that one view alone does not provide enough information for complete 3-D reconstruction because a single camera produces only two-dimensional (2-D) measurements of 3-D scene structure. In this paper, we describe a real-time camera tracking framework designed to track a monocular camera in an AR workspace. In fact, the use of a projective camera as a primary sensor introduces additional difficulties to the problem. Tracking a hand-held camera is more difficult than tracking a specific device, such as a robot. In case of a robot, it usually receives a sort of odometry data and can be driven at arbitrarily slow and controllable speed.[1] Acquisition of landmark position is also easier than that of monocular visual tracking because, in a robot, actual positions of landmarks are received directly from a series of specific sensors. This is not the case for monocular visual tracking. A single camera is a bearing-only sensor and it provides only 2-D measurements of 3-D scene space. In this paper, we focus on integration of a bundle of nonlinear filters to achieve robust camera tracking and scalable feature mapping, which can extend to larger environment. The main contribution of the proposed work is that a particle filter–based camera tracker is combined with independent feature estimators, which have fixed-state dimension. In addition, we split camera tracking and feature estimation into two separate tasks, which are handled in two parallel working pipelines. Therefore, the camera pose estimation and feature tracking are handled independently, so the camera tracking is less influenced by increase of the number of features. An additional feature of our system is that it employs predictive template warping in order to minimize problems caused by straightforward template matching, which is quite vulnerable to unpredictable camera motions. This approach may seem to be similar to the method introduced in Refs. 2 and 3. However, our method is quite different in that every estimate required for the predictive template warping can be obtained directly from the independent feature estimators, so that the template prediction is maintained without additional framework for template state estimation.

## 2 Related Works

The use of a monocular camera introduces additional difficulties, and therefore, the filtering techniques that enable camera state acquisition from prediction-correction model are very important to the problem. The majority of conventional approaches that support the prediction-correction model have been based on variants of the Kalman

*Address all correspondence to: Seok-Han Lee, E-mail: seokhan@jj.ac.kr

filter.[2–4] In particular, the extended Kalman filter (EKF) linearizes the state model of system and represents all distributions as Gaussians. So a plurality of tracking systems is achieved on the basis of the EKF framework. In contrast to conventional structure-from-motion (SFM) techniques, which rely on global nonlinear optimization, recursive estimation methods permit online operation, which is highly desirable for a real-time camera tracking system. For example, Davison shows the feasibility of real-time simultaneous localization and mapping with a single camera in Ref. 2, using the EKF estimation framework. His system takes a top-down Bayesian estimation approach, searching for features in image regions constrained by state uncertainty instead of performing bottom-up image processing and feature matching. Efforts to improve the robustness of camera tracking have recently been made by Pupilli and Calway,[5] who replace typical EKF framework with particle filtering, which is quite robust to erratic camera motions. However, the goal of the work is on robust camera tracking, and new feature mapping is not discussed. In Refs. 6 and 7, FastSLAM-style particle filters for the real-time camera tracking are presented. The camera tracking is performed by particle filtering, while feature mapping is achieved by a series of estimators that are included in each particle. FastSLAM framework has the advantage of scalability, although the computational cost may prohibit real-time operation given a large number of particle sets and feature points. Our approach is different in that a single map is maintained by independent unscented Kalman filters (UKFs) while the camera pose is estimated by a set of particle cloud. An alternative approach is taken by Chekhlov et al.,[8,9] who employ a more robust image descriptor instead of conventional correlation-based scheme, which may reduce the probabilities of outlier measurements. This allows the system to operate with large regions for feature search without compromising robustness. Camera tracking based on batch techniques has previously been introduced by Genc et al. and Subbarao et al.[10,11] A specific tracking system or fiducial markers are used in an initialization phase to estimate new feature points, which can later be employed for the tracking. In particular, Ref. 10 uses conventional bundle adjustment technique in the training phase and presents decent tracking performance when tracking the estimated reference features. However, no attempt is made to extend the feature map after the initialization stage. Reference 11 presents an algorithm for more robust and accurate tracking; however, this method might cause a severe performance degradation, which may lead the system operation to unusable levels. Moreover, it is not clear if the feature map can grow after the initialization. References 12 and 13 present a framework for real-time degrees of freedom (6DOF) camera pose estimation based on natural features in an arbitrary scene. Crucially, the systems are operated on the basis of precaptured reference images. Feature points in the current image frame are matched to two spatially separated reference images. In Ref. 12, this wide baseline correspondence problem is solved by computing a global homography between current and previous image frame and employ local affine assumption between previous frame and reference images. Although this system is claimed to be more robust and accurate, any discussion on the scalability of feature map or erratic camera motion is not given. In Ref. 14, a particle filtering framework for real-time camera pose estimation is presented. In particular, the authors propose an approach for real-time points and line tracking to estimate camera pose, and demonstrate how to use the 3-D line constraint in order to construct likelihood function for particle filtering. The camera's pose is assumed to undertake the first-order random-walk with uniform uncertainty in the state space. This system can track camera accurately under various situations, such as severe occlusion and nonsmooth camera motion. However, this work focuses only on robust camera tracking,

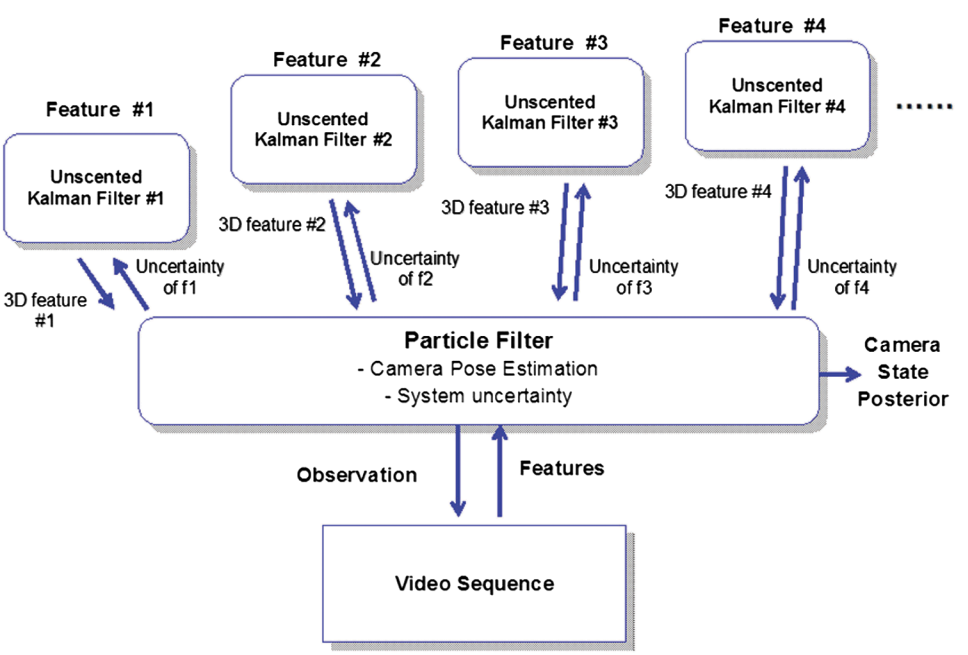


**Fig. 1** Block diagram of the proposed framework. A particle filter is connected to independent filters. The camera tracking is achieved by the particle filter, while the feature tracking is performed by independent feature estimators, which have a fixed dimension.

and any discussion on new feature mapping and scalability of feature mapping is not presented. Reference 15 proposes a camera tracking method for multiview sequential images, such as aerial video stream, which exploits the constraints imposed by the path of camera motion to allow detection and correction of inaccuracies in the feature tracking and structure computation processes. The camera path is used as a source of strong constraints in feature tracking, such that tracks that do not meet the constraints can be corrected, resulting in an improved scene structure. This approach improves the accuracy of feature tracking and, therefore, allows quite robust 3-D reconstruction. But this method is basically for off-line SFM framework, so any discussion on real-time operation is not given. Reference 16 illustrates a system for real-time camera tracking and reconstruction, which relies not on feature extraction technique but dense, every pixel methods. This system builds a set of keyframes from camera image stream and estimates detailed textured depth maps at selected keyframes to produce a surface patchwork that has millions of vertices. In order to improve the quality of the final output, the system uses the hundreds of images available in a video stream. This system achieves real-time performance using commodity GPU hardware and produces an impressive volumetric depth map reconstruction of a scene with high accuracy. Reference 17 proposes a technique to extend 3-D map-building to allow one or more cameras to work in several maps, separately or simultaneously. In fact, this allows spatially separated AR work spaces to be constructed and used without user's intervention. In order to achieve this, it exploits the uniform and dense spread of keyframes through a map. A descriptor to detect each keyframe is built by conventional image subsampling technique and Gaussian blur. And current camera image is compared to those from keyframes in order to set the camera's pose to that of the keyframe. In order to resolve the multiple map problem, the relocalization mechanism cross-correlates with all of the keyframes from all maps. In addition to the above approaches, several research efforts have been devoted to make use of sensor fusion and more diverse visual features in Refs. 18 and 19.

Recently, the advent of RGB-D sensors like the Microsoft Kinect opened new possibilities for approaches on dense 3-D reconstruction as they directly output dense depth information. The KinectFusion algorithm introduced by Newcombe et al.[20] was one of the first systems to produce a volumetric reconstruction of a scene in real time with very high accuracy. In particular, they have presented impressive results by using signed distance functions to represent the scene geometry and the iterated closest point algorithm for camera tracking. References 21 and 22 present extended versions of the original KinectFusion algorithm, which aim to boost the performance and accuracy of the original algorithm. These methods combine diverse odometry estimation algorithms in order to increase the robustness and accuracy of camera tracking across a variety of environments, from desktop environments to extended scale paths. By virtue of depth information directly received from the sensor, these techniques can achieve very accurate camera tracking and high-quality volumetric reconstruction. Efforts to improve the robustness of relocalization, which is known as loop-closure problem, have recently been made by Martinez-Carranza and Mayol-Cuevas.[23] Binary descriptors combined with a hashing technique are employed, and the depth information available from an RGB-D sensor is used to assist 3-D geometric validation stage in the selection of good feature sample matches. In fact, the loop-closure problem is one of the main challenges in robots equipped with vision sensor. The authors show that depth information from the RGB-D camera can significantly improve the accuracy. Reference 24 illustrates how the RGB-D cameras can be used specifically for building dense 3-D maps of indoor environments. The authors employ an optimization algorithm combining visual features and shape-based alignment, and present a full 3-D mapping system. In order to achieve

**Algorithm 1** Particle filtering for camera pose estimation

**Data at time step** $k$: $\mathbf{S}_{k-1}$, $\mathbf{V}_C$, $\Omega_C$, $I_k$

**Output**: $\mathbf{S}_k$ (new particle set)

{**Step 1.** resample particle set}

$\mathbf{S}_{k-1}^R = \text{Resample}(\mathbf{S}_{k-1})$;

{**Step 2.** apply motion model to all particles}

$\mathbf{S}_k^- = \varnothing; \mathbf{M}_k^- = \varnothing$;

**for** $(\mathbf{x}_{k-1}^n \in \mathbf{S}_{k-1}^R)$

$\mathbf{x}_k^n = f\ (\mathbf{x}_{k-1}^n, \mathbf{V}_C, \Omega_C)$; // *apply motion model*

add $\mathbf{x}_k^n$ to $\mathbf{S}_k^-$;

add feature projections $\mathbf{m}_k^n = P(\mathbf{x}_k^n) \cdot \mathbf{z}_m$to $\mathbf{M}_k^-$;

**end for**

{**Step 3.** compute feature measurements}

compute search regions based on $\mathbf{M}_k^-$;

**for** $(\mathbf{z}_m \in Z)$

$\mathbf{r}_{k,m}(\mathbf{z}_m) = T(\mathbf{z}_m) * I_k$; // NCC & template matching of $\mathbf{z}_m$

$\mathbf{y}_{k,m} = \text{Threshold} < \mathbf{r}_{k,m}(\mathbf{z}_m)$; // obtain measurements

**end for**

{**Step 4.** compute particle weights & form $\mathbf{S}_k$}

$\mathbf{S}_k = \varnothing$;

**for** $(\mathbf{x}_k^n \in \mathbf{S}_k^-)$

$w_k^n = p(y_k | \mathbf{x}_k^n, Z)$;

build $(\mathbf{x}_k^n, w_k^n)$ & add it to $\mathbf{S}_k$;

**end for**

{**Step 5.** normalize the particle weights of $\mathbf{S}_k$}

Normalize($S_k$);

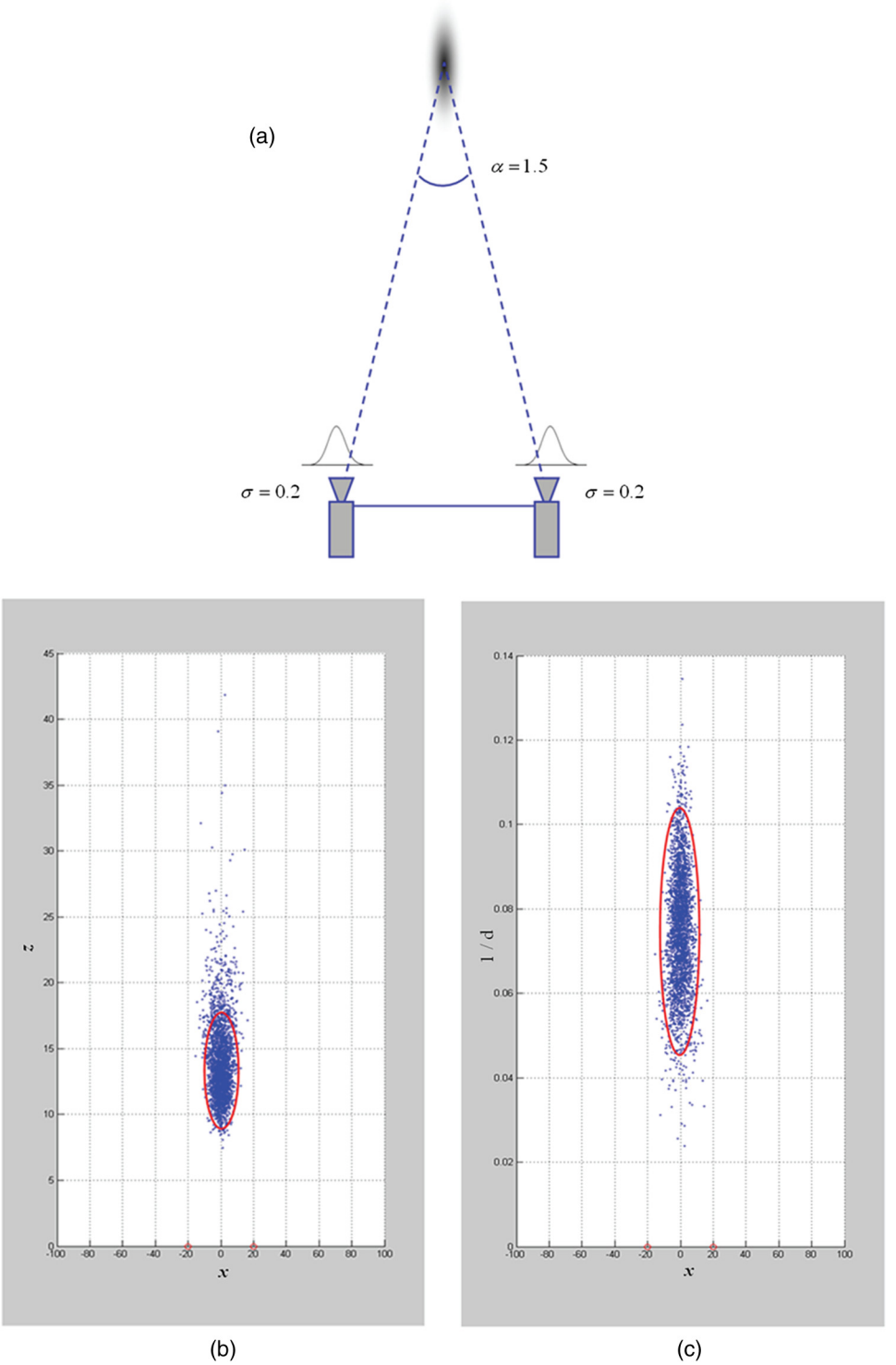


**Fig. 2** Simulation of the inverse depth estimation. Red ellipses represent depth uncertainties of the Gaussian error of projection rays: (a) Simulation condition for two-dimensional point estimation, (b) point reconstruction modeled with xyz coordinate, and (c) with inverse depth, respectively.

this, visual and depth information are combined for view-based loop closure detection, followed by camera pose optimization to achieve globally consistent maps.

In this paper, our goal is to build a framework that can manage many features simultaneously in real time while estimating camera pose robustly. As illustrated in Fig. 1, basic idea of the proposed method is that a particle filter-based camera tracking is connected to multiple feature estimators via measurement covariances. In particular, the camera tracking is achieved by a main particle filter, which deals with the task of camera tracking, while the feature mapping is performed by independent feature estimators, which have fixed dimension. The proposed framework has the following advantages. By virtue of the particle filtering, our system presents robust camera tracking, which can cope with unpredictable camera motions, such as rapid camera shake or

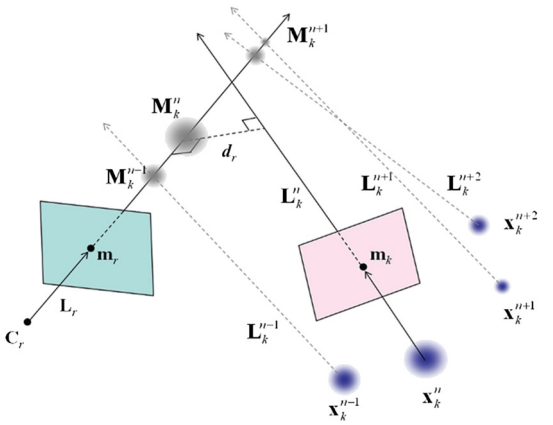


**Fig. 3** Initial depth estimation based on the reference frame $C_r$ and current camera particle set $\mathbf{x}_k$. Blue spheres represent camera particles, and their sizes mean weights based on the normal distribution, which is parameterized by distance $p_r$ between each projection ray $L_k$ and corresponding intersection point $M_k$.

occlusion. In addition, new features are tracked by independent UKFs with fixed dimension, and more features can be managed simultaneously. These allow our system to scale to larger environment while achieving robust camera tracking. An additional feature of our system is that every estimate required for predictive warping of feature template is obtained from independent UKFs so that template prediction can be maintained without any additional filters for the template state estimation. Moreover, we split the camera tracking and feature mapping into two separate tasks, which are handled in two parallel processes. This approach shares the theoretical basis presented in previous work, such as Ref. 25. However, our system is different in that we split camera tracking and feature tracking into two separate processes and implement an incremental mapping approach in which a sparse map of relatively high-quality features are maintained by feature estimators linked to the particle filter. We verify the effectiveness of the proposed approach operating under an AR environment.

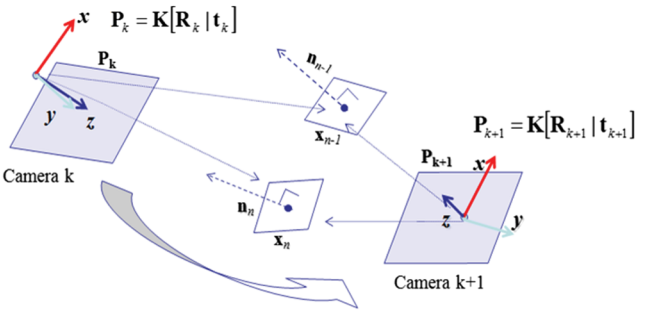


**Fig. 5** Geometry of template prediction. Feature templates are warped by projective mapping between the camera projection matrix of time step $k$ and $k+1$.

## 3 Proposed Framework

### 3.1 *Camera Tracking Using a Particle Filter*

For the robust camera tracking, we recursively estimate the pose (position and orientation) of camera for each time step $k$. We use a state space model, which has the form $\mathbf{x}_k = [\mathbf{t}_k, \mathbf{q}_k]^{\mathrm{T}}$, consisting of a $3 \times 1$ translation vector $\mathbf{t} = [\, t_x \quad t_y \quad t_z \,]$ and a $4 \times 1$ quaternion (rotation) vector $\mathbf{q} = [\, q_0 \quad q_x \quad q_y \quad q_z \,]$ to parameterize the camera's state with respect to the world coordinate system at time step $k$. It is assumed that we have a set of 3-D points $Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_{M-1}, \mathbf{z}_M]$ whose locations are given in the world coordinate frame, where $M$ is the number of 3-D points and $\mathbf{z}_m$ the $m$'th 3-D point. For the camera state $\mathbf{x}_k$, we can compute a projection of the $m$'th 3-D feature point $\mathbf{z}_m$ using the 3-D–2-D projective mapping of the camera as follows:

$$\mathbf{y}_k = P(\mathbf{x}_k) \cdot \mathbf{z}_m = \mathbf{K}[R(\mathbf{q}_k) \cdot \mathbf{z}_m + \mathbf{t}_k] = \mathbf{K}[\mathbf{R}_k|\mathbf{t}_k]\tilde{\mathbf{z}}_m. \quad (1)$$

Here, $\mathbf{K}$ is the $3 \times 3$ camera calibration matrix and $R(\mathbf{q})$ a $3 \times 3$ rotation matrix of a quaternion vector $\mathbf{q}$. Symbol "$\sim$" denotes the homogeneous coordinate representation. The camera tracking is then formulated as determining the posterior, $p(\mathbf{x}_k|\mathbf{y}_{1:k}, Z)$, given the 3-D feature points and their 2-D projections $\mathbf{y}_{1:k} = [\mathbf{y}_1, \ldots \mathbf{y}_k]$, which denotes a set of

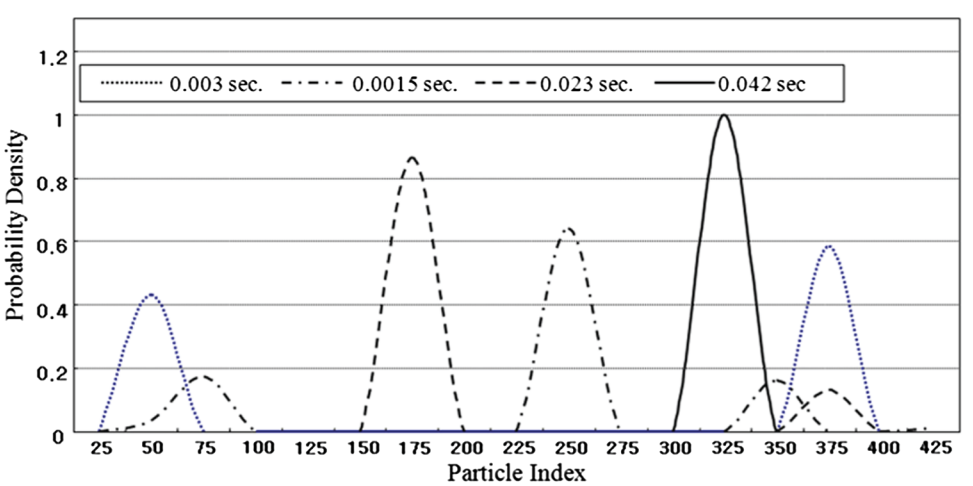


**Fig. 4** Convergence of initial depth likelihood for successive observations for a new feature. Note that multimodal and highly non-Gaussian depth likelihood estimate converges to a Gaussian-like unimodal shape (solid line) within 50 ms.


**Fig. 6** Observed feature templates and predictive template warping. Note that respective templates are warped according to change of the camera pose.

measurements obtained up until the time step $k$. At each time step, we can take measurements from input image and obtain successive estimates from a recursive Bayesian filter. Assuming that 3-D scene structure is static and independent of the system estimates, the recursive process is given as follows:[26]

$$p(\mathbf{x}_k|\mathbf{y}_{1:k},Z)=\frac{p(\mathbf{y}_k|\mathbf{x}_k,Z)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}. \tag{2}$$

This equation is defined by the measurement likelihood $p(\mathbf{y}_k|\mathbf{x}_k,Z)$ up to the current time step, and $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$, which incorporates the probability about the camera motion. The camera tracking involves obtaining successive approximations to the posterior $p(\mathbf{x}_k|\mathbf{y}_{1:k-1},Z)$. The particle filter can provide this information in a form of weighted sample particle set $\mathbf{S}_k=[(\mathbf{x}_k^1,w_k^1),(\mathbf{x}_k^2,w_k^2),\ldots,(\mathbf{x}_k^N,w_k^N)]$. Here, $\mathbf{x}_k^n$ is the $n$'th sample of $N$ camera particles at time step $k$; its weight $w_k^n$ is proportional to the conditional likelihood $p(\mathbf{y}_k|\mathbf{x}_k,Z)$. In order to obtain the state estimate, we need a probabilistic model for the state transition between time steps, i.e., $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, and a likelihood $p(\mathbf{y}_k|\mathbf{x}_k,Z)$ for the observations given the state and the scene structure. We currently employ a constant position model, which is more general and compact than a constant velocity or constant acceleration assumption.[27,28]

$$\mathbf{x}_k=\begin{pmatrix}\mathbf{t}_k\\ \mathbf{q}_k\end{pmatrix}=\begin{pmatrix}\mathbf{t}_{k-1}+\mathbf{V}_c\\ \mathbf{q}_{k-1}\otimes\mathbf{q}(\Omega_c)\end{pmatrix}, \tag{3}$$

where $\mathbf{V}_c$ and $\Omega_c$ are Gaussian uncertainties for the camera position and the angular displacement, respectively. $c$ represents the camera coordinate system, $\otimes$ denotes quaternion product operation. Equation (3) states that the camera is assumed to be in the same place from one frame to the next, but it has Gaussian and unimodal uncertainty, which in effect allows the camera's pose to undertake a first-order random-walk in the state space. This constant position model has a nonadditive noise model due to the quaternion product, and this assumption often causes critical problems within variants of the Kalman filter. An advantage of the particle filter approach is that it does not matter whether the

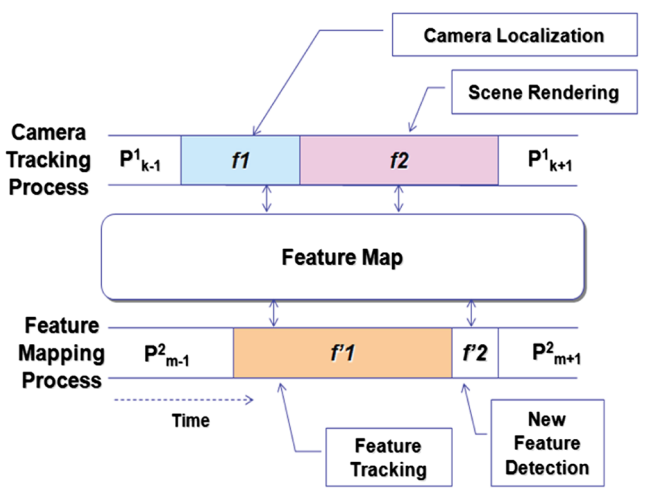


**Fig. 7** Parallelization of camera pose estimation and feature tracking. Two processes are separated into two tasks, so that the camera tracking is less influenced by increase of the number of features.

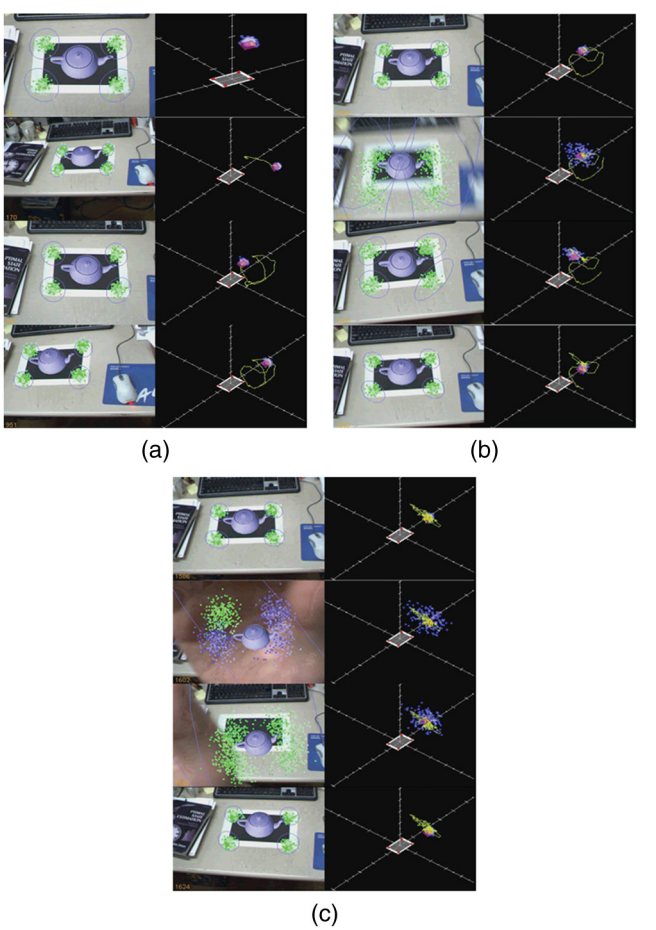


**Fig. 8** Tracking results of the proposed system for (a) stable condition, (b) rapid camera shaking, and (c) occlusion. (Video 1, QuickTime, 8.54 MB) [URL: http://dx.doi.org/10.1117/1.JEI.23.1.013029.1].

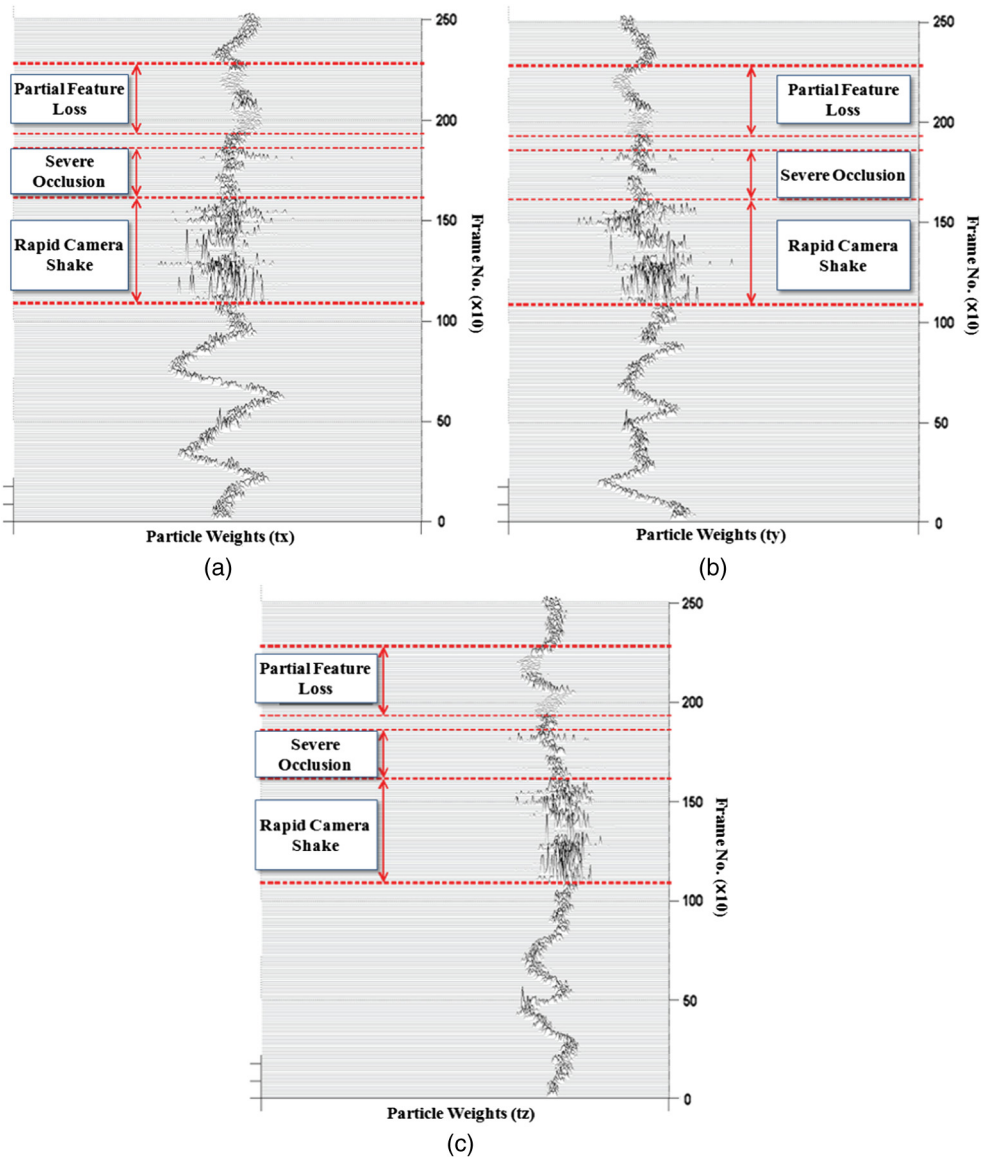


**Fig. 9** Posterior distribution for each of the camera position parameters over the tracking sequence: (a) $t_x$, (b) $t_y$, and (c) $t_z$.

noise model is additive or nonlinear. Therefore, we can simply draw a noise component from a Gaussian distribution for each particle and compute the nonlinear constant position function. The next state of the camera pose is defined by $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and can be modeled by a Gaussian distribution

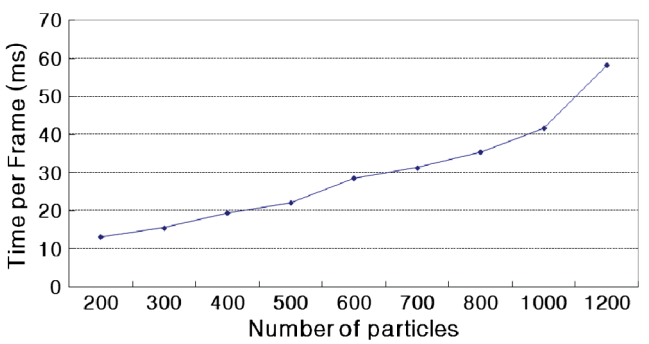


**Fig. 10** Time consumption for one frame against number of particles and 15 features.

$p(\mathbf{x}_k|\mathbf{x}_{k-1}) \propto N(\mathbf{x}_{k-1}, \Sigma)$, which has mean $\mathbf{x}_{k-1}$ and covariance $\Sigma$. The likelihood $p(\mathbf{y}_k|\mathbf{x}_k, Z)$ is determined based on the Gaussian distribution parameterized by the distance between measurements of the points in $Z$ and projections of $\mathbf{z}_m$ by the camera particles. From the $N$ sample particles and the relationship of Eq. (1), we obtain a set of projections corresponding to the $i$'th feature point, that is, $M_i = [\mathbf{m}_i^1, \mathbf{m}_i^2, \ldots, \mathbf{m}_i^{N-1}, \mathbf{m}_i^N]$. This set constitutes projected particle cloud, which involves the probability distribution about current state. Given a particle $\mathbf{x}_k^n$, the $n$'th particle in the frame $k$, the relative likelihood $w_k^n$ that $\mathbf{m}_i^n$ is equal to $\mathbf{y}_{k,i}$, can be computed as follows:

$$w_k^n = p(\mathbf{y}_k|\mathbf{x}_k^n, Z) \propto \frac{1}{2\pi \cdot |\mathbf{C}|^{1/2}} \cdot \exp\left[-\sum_{i=1}^{L} (\mathbf{m}_i^n - \mathbf{y}_{k,i})^{\mathrm{T}} \mathbf{C}^{-1} (\mathbf{m}_i^n - \mathbf{y}_{k,i})/2\right], \quad (4)$$

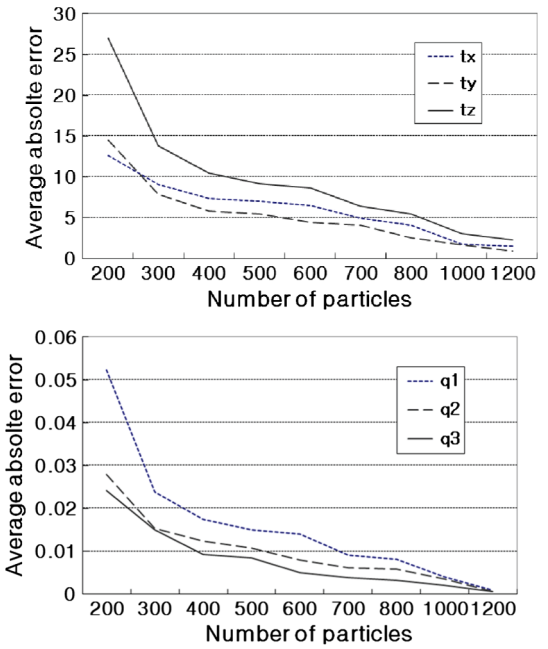


**Fig. 11** Tracking accuracy against number of particles. Note that tracking errors increase as the number of particles decreases: (a) position error and (b) quaternion error.

where $\mathbf{L}$ is the total number of features in the camera image and $\mathbf{C}$ the covariance matrix, which can be obtained from the distribution of the projected particle cloud. Equation (4) represents that camera particles with feature measurements close to real observations will be given higher weight. In order to obtain the observation $\mathbf{yk,i}$, we compute a set of correlations $\mathbf{rk,i} = \mathbf{T(zi)} * \mathbf{Ik}$, one for each feature point. Here, * means the normalized cross-correlation operation, $\mathbf{Ik}$ is an image frame of the time step $\mathbf{k}$, and $\mathbf{T(zi)}$ denotes an image template of the $\mathbf{i}$'th scene point $\mathbf{zi}$. The templates are then correlated with subsequent input images to indicate potential corresponding feature points. The relative likelihoods obtained in Eq. (4) are normalized so that $\sum_{n=1}^{N} w_i^n = 1$, and this ensures that the sum of all the likelihoods is equal to 1. We resample the particles from the computed likelihood so that new particles are randomly generated based on the relative likelihood $w_k^n$ and the density $\mathbf{p}(\mathbf{x}_k|\mathbf{x}_{k-1})$, and obtain a set of particles that are distributed according to $\mathbf{p}(\mathbf{y}_k|\mathbf{x}_k, \mathbf{Z})$. The camera pose is obtained from the mean of particles. The proposed particle filtering for camera tracking is detailed in pseudocode of Algorithm 1.

### 3.2 *New Feature Detection and Proposed Parameterization Method*

The camera tracking framework relies on the knowledge of 3-D features $\mathbf{Z}$, and they need to be estimated simultaneously with the camera state. However, the system has no prior knowledge about the scene structure when it is first switched on.

We initialize the camera state using a rectangular reference pattern, which provides four corner features with known positions and known appearances. The task then is to dynamically detect new features and to recursively estimate their corresponding coordinates. New features are detected by running the FAST corner detector to find the best candidate within a search box of predefined size placed within the image.[29,30] The position of the search box is chosen randomly, with the constraint that it should not overlap with any existing features. For the detected features, we estimate their depths using a set of UKFs, which take the role of auxiliary feature estimators. One problem of the UKF-based system is that its computational cost increases by $N^5$ for $N$ feature points. Since the number of features and the system dimension grow with time, the computational cost quickly exceeds the maximum capacity allowed for the real-time operation. In our system, we proposed a method to estimate the feature coordinates with independent estimators. In particular, we employ UKFs that are independently assigned to each of the feature points to track. Therefore, we have $N$ independent UKFs corresponding to each of the $N$ feature points. As each independent filter has a fixed-state dimension, computational cost of each filter does not grow as in the case of the monolithic UKF framework, and therefore, entire processing time required for the feature mappings can be considerably reduced. We parameterize the location of a feature in terms of its direction and depth from the camera, and they are defined by a unit vector $\mathbf{v}_r$ in the direction of the feature and depth $d_r$ in the direction of the projection ray, respectively. Here, $r$ represents the reference time when the new feature is first detected. For each new feature, $\mathbf{v}_r$ can be computed from the camera center and the feature's location in the image coordinate system. But the depth of the feature $d_r$ is not certain. Therefore, this should be included in the feature state estimation. Typically, the depth information is parameterized in terms of its inverse form since it makes estimation more linear and Gaussian. Moreover, it is also more efficient for representing very large numbers and, therefore, allows feature points to be mapped over larger ranges.[31] Figure 2 shows the simulation

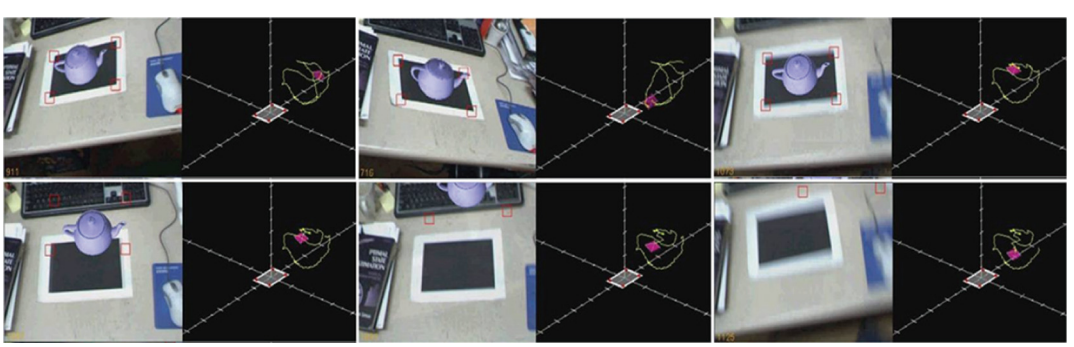

**Fig. 12** Tracking failure of unscented Kalman filter (UKF) for rapid camera shake. (Video 2, QuickTime, 4.7 MB) [URL: http://dx.doi.org/10.1117/1.JEI.23.1.013029.2].

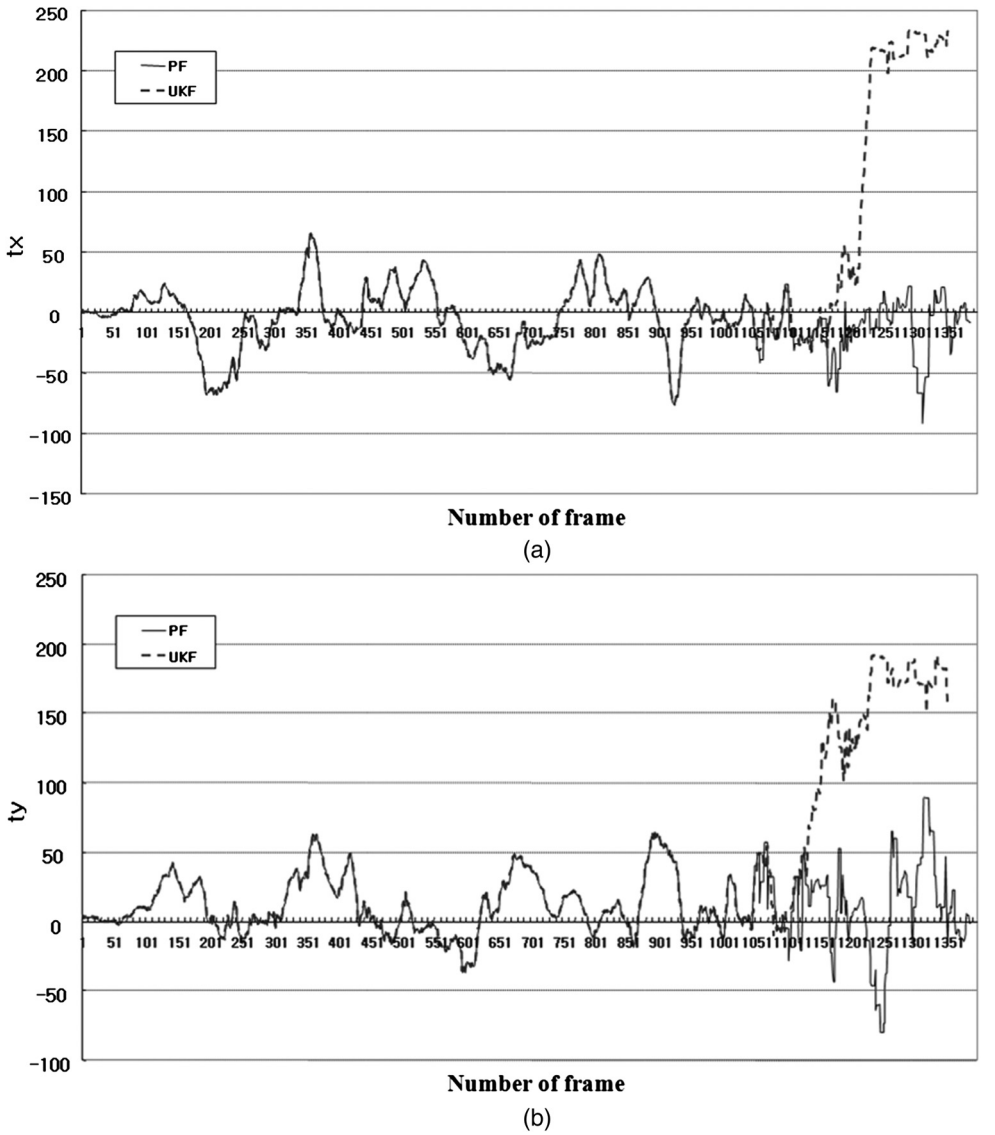


**Fig. 13** Camera state estimates obtained from the particle filter and the UKF. (a) $t_x$ and (b) $t_y$.

results, which illustrate a point reconstruction from two parallax observations when observed by two cameras at known locations. It is not difficult to verify that the reconstruction result using the inverse depth is even closer to the Gaussian distribution. As the camera pose and 2-D location of the new feature are also uncertain, they need to be included in the state estimation. From this state model, we can compute coordinate of a feature point in the world coordinate system. Once we detect a new feature, we initialize the UKF mean and covariance as follows:

**Table 1** Performance comparison of tracking approaches.

| Methods | Filter type | Computation time (ms) | Registration error (MSE) |
|---|---|---|---|
| 14 | Particle filter | 18 | 0.98 |
| 5 | Particle filter | 17.52 | 0.89 |
| UKF | Unscented Kalman filter | 7.03 | 0.66 |
| Proposed system | Particle filter | 10.12 | 0.87 |

Note: MSE, mean squared error.

$$\mathbf{c}_0 = [\bar{\mathbf{t}}_r, \bar{\mathbf{q}}_r, 1/\bar{d}_r, \mathbf{m}_r^{\mathrm{T}}], \mathbf{m}_r^{\mathrm{T}} = (x_r, y_r)^{\mathrm{T}}, \tag{5}$$

$$\Sigma_0 = \begin{bmatrix} \Sigma_{Cr} & \cdots & \mathbf{0} \\ & \Sigma_{\mathrm{inv}dr} & \\ \mathbf{0} & \cdots & \Sigma_{xr} \end{bmatrix}, \Sigma_{\mathrm{inv}dr} = \sigma_{\mathrm{inv}dr}^2, \Sigma_{xr} = \begin{bmatrix} \sigma_{xxr}^2 & \sigma_{xyr}^2 \\ \sigma_{yxr}^2 & \sigma_{yyr}^2 \end{bmatrix}. \tag{6}$$

Here $\Sigma_{Cr}$ is a $7 \times 7$ covariance of the camera state vector $\mathbf{x}_r = [\mathbf{t}_r, \mathbf{q}_r]$, and $\Sigma_{xr}$ is a $2 \times 2$ covariance matrix of the new

**Table 2** Time consumption of each processing step (ms).

| | Process | 10 features | 30 features | 50 features | 70 features | 90 features |
|---|---|---|---|---|---|---|
| 1st process | New particle draw & measurement samples | 6.8 | 9.1 | 12.51 | 14.99 | 16.4 |
| | Feature tracking & particle priori estimation | 5.1 | 6.78 | 7.38 | 8.04 | 8.61 |
| | Particle resampling & camera state estimation | 0.12 | 0.25 | 0.34 | 0.41 | 0.45 |
| | Rendering | 5.1 | 5.33 | 5.41 | 5.48 | 5.52 |
| | Total | 17.12 | 21.46 | 25.64 | 28.92 | 30.98 |
| 2nd process | New feature incoorporation & map management | 3.81 | 4.04 | 4.25 | 4.78 | 5.03 |
| | New feature detection & verification | 0.23 | 0.25 | 0.29 | 0.32 | 0.41 |
| | UKFs | 11.11 | 13.07 | 15.99 | 17.01 | 21.88 |
| | Total | 15.15 | 17.36 | 20.53 | 22.11 | 27.32 |

feature point of reference image in which the new feature is first detected. $\Sigma_{xr}$ can be obtained from the distribution of the projected particle set. $\Sigma_{invdr}$ is a variance of the inverse depth, which is distributed along the projection ray.

### 3.3 *Proposed Method for Initialization of New Feature Point*

At the time when a new feature is detected, we do not have any information on correlation between the camera pose and state of the new feature. Therefore, other elements of $\Sigma_0$ in Eq. (6) are initially set to zero. The camera mean $[\bar{\mathbf{t}}_r, \bar{\mathbf{q}}_r]$ and covariance $\Sigma_{Cr}$ for the reference frame are computed from the particle set $\mathbf{S}_k = [(\mathbf{x}_k^1, w_k^1), (\mathbf{x}_k^2, w_k^2), \ldots, (\mathbf{x}_k^N, w_k^N)]$. At this point, we have no information on the depth of new feature, so the initial depth and the variance $\Sigma_{invdr}$ cannot be initialized directly. In this paper, we propose a method to compute the initial depth and its variance as follows. Once a new feature point is detected, its 2-D coordinate, template image, and the camera pose at the time when the feature is first detected are recorded. The template is then correlated with subsequent images to indicate potential corresponding points. Since we have the camera pose in both the initial (reference) and subsequent frames, we can triangulate potential corresponding points to obtain depth distribution of the new feature point, as illustrated in Fig. 3. Given the particle distribution of the camera states in time step $k$, represented as the particle set $\mathbf{S}_k$, we can compute a number of back-projection rays $\mathbf{L}_k^n$ corresponding to each of the sample particles $\mathbf{x}_k^n$ for the candidate feature template. Each of these rays intersects with the back-projection ray $\mathbf{L}_r$ of the reference image frame. If the two back-projection rays intersect perfectly, the distance between the two rays will be zero. In general, however, they are not likely to meet at one intersection point perfectly. In this case, the intersection is given as a point $\mathbf{M}_k^n$, where the distance between $\mathbf{L}_r$ and $\mathbf{L}_k^n$ becomes minimum. We assign a weight to each intersection point based on the normal distribution parameterized by the distance between the rays ($p_r$ of Fig. 3). Finally, we obtain a one-dimensional depth distribution lying along the reference ray $\mathbf{L}_r$. This distribution is recursively updated over a number of frames. Once this distribution becomes unimodal, we assume that the distribution is approximately Gaussian and compute the mean of inverse depth and its variance, and hand the estimates over to the feature estimator. This process continues until the weight distribution converges to a unimodal form. If the distribution does not converge during a predetermined number of matching attempts, it is not regarded as a true 3-D feature, and the feature initialization is abandoned. Figure 4 shows evolution of the distribution over time. We can verify that the distribution converges quickly on a Gaussian form as the camera moves. From the initial mean and covariance given as Eqs. (5) and (6), the 3-D coordinate of the $m$'th feature in the $k$'th frame is computed as follows:

$$\hat{\mathbf{z}}_m = \mathbf{t}_{km} + \frac{1}{d_{km}}\mathbf{D}_{km}, \mathbf{D}_{km} = (\bar{\mathbf{P}}_k)^{-1}\tilde{\mathbf{m}}_{km}, P(\bar{\mathbf{x}}_k) = [\bar{\mathbf{P}}_k \tilde{\mathbf{p}}_k]. \tag{7}$$

Here, $\mathbf{D}_{km}$ is the direction of the $m$'th feature point, $\mathbf{t}_{km}$ is the camera position corresponding to the $m$'th feature in the $k$'th frame, and $\bar{\mathbf{P}}_k$ represents the $3 \times 3$ submatrix of the camera projection matrix of Eq. (1), that is, $\bar{\mathbf{P}}_k = \mathbf{K} \cdot \mathbf{R}_k$. And $d_{km}$ is the depth of feature point.

### 3.4 *Feature Tracking Using UKF*

In the proposed framework, we assume that the scene is static, and a nonlinear state equation is defined by $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{n}_k)$. Here $\mathbf{n}_k$ is a system noise vector and is assumed to be a zero vector in the proposed framework. For feature points $\mathbf{y}_{mk}$, the $m$'th feature point of the $k$'th frame, we define a measurement model as follows:

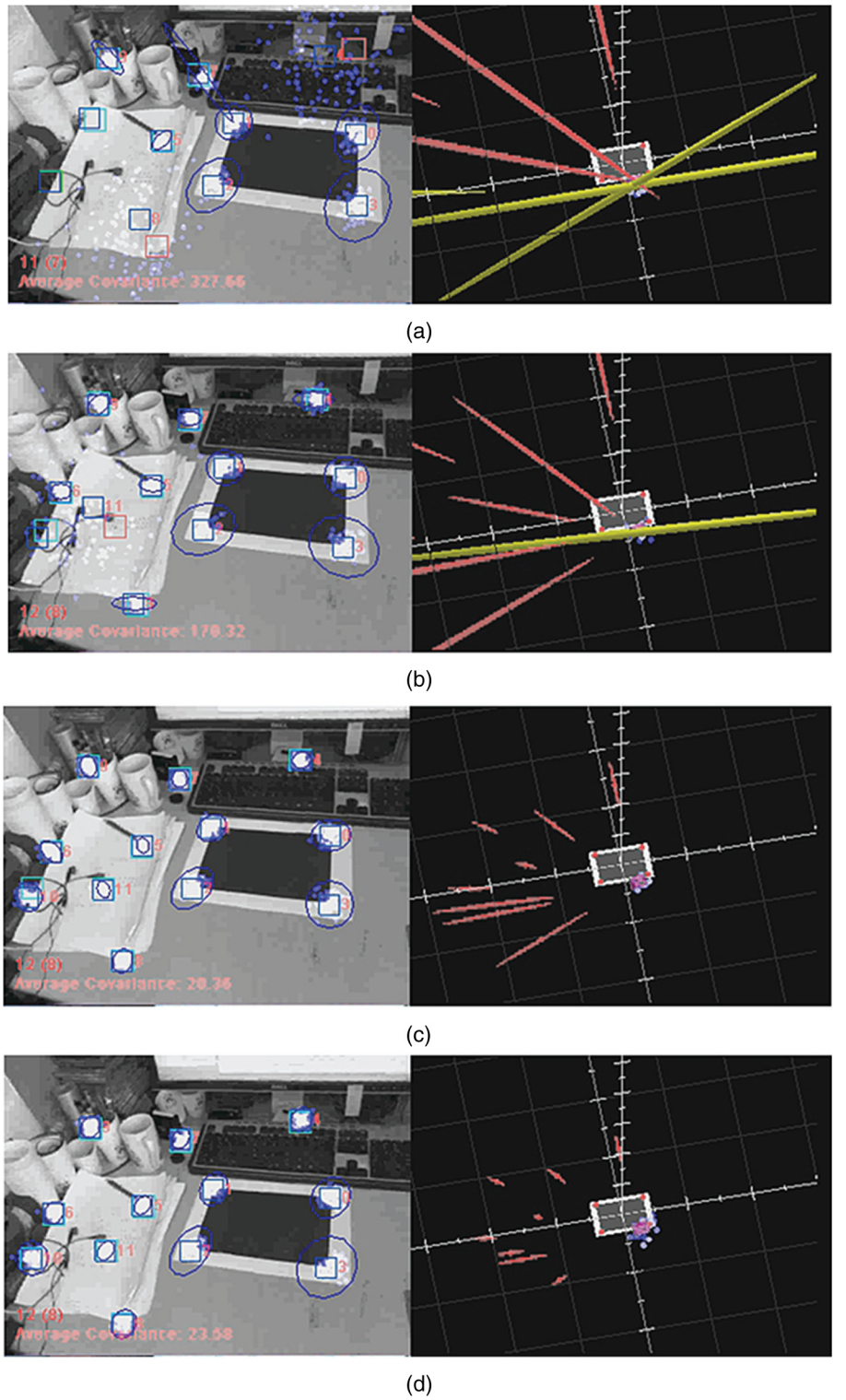


**Fig. 14** Initializations of feature points: (a) frame #422, (b) #429, (c) #435, and (d) #441.

$$\mathbf{y}_{mk} = [P(\bar{\mathbf{x}}_k)\mathbf{z}_m] + \mathbf{n}_{mk} = \mathbf{K}[\bar{\mathbf{R}}_k^n|\bar{\mathbf{t}}_k^n]\mathbf{z}_m + \mathbf{n}_{mk} = \begin{bmatrix} x_{mk} \\ y_{mk} \end{bmatrix} + \mathbf{n}_{mk}. \quad (8)$$

Here, $\mathbf{n}_{mk} \in R^{2M}$ is a noise vector for the $m$'th feature point of the $k$'th frame, and its covariance matrix is given as follows:

$$\mathbf{R}_k = \mathbf{R}_p + \sigma_{xk}^2 = \mathbf{R}_p + \mathrm{Cov}[P(\bar{\mathbf{x}}_k)\mathbf{z}_m]. \quad (9)$$

In Eq. (9), $\mathbf{R}_p$ is due to the image pixel noise and can be assumed to be very small. $\sigma_{xk}^2$ represents the uncertainty of current camera state $\mathbf{x}_k$. This measurement noise covariance is computed for every frame $k$ by projecting each feature point $\mathbf{z}_m$ into the camera particles. If the feature tracking is accurate, the estimated feature coordinate will be

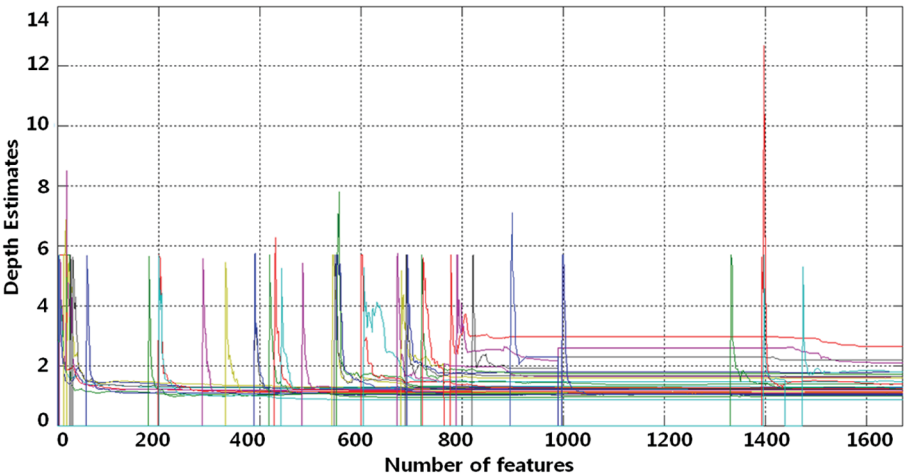


**Fig. 15** Initial depths of feature points converging to depth estimates.

**Table 3** Average feature estimation difference between particle filter (PF) and UKF.

| Feature No. | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 | Avg. Diff. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PF | 0.00 | 0.00 | 0.00 | 0.00 | −1.15 | 1.33 | −1.25 | 3.00 | 2.25 | −1.21 | 0.50 |
| UKF | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | −0.70 | 0.59 | −1.80 | 0.53 | 0.46 | 0.01 |
| \|PF-UKF\| | 0.00 | 0.00 | 0.00 | 0.00 | 2.15 | 2.03 | 1.84 | 4.80 | 1.72 | 1.67 | 2.37 |

coincident with the real feature location. In this case, feature projections into the camera particles distribute near the correct feature point, and they constitute a Gaussian distribution. In the case of a wrong feature, however, the projection point is not likely to be located near the correct point, and its probability tends to approach zero. Any features that do not converge rapidly to the correct position are regarded as wrong features and removed from the map.

### 3.5 *Predictive Warping of Feature Template*

As described in the previous section, new feature templates are detected by using the FAST corner detection operator from the images obtained from the camera.[29,30] Then the template matching attempts to identify the same feature template from subsequent image frames. However, straightforward template matching may be limited under unpredictable camera motions, as even by a small change of camera pose the appearance of a template can be greatly changed. In order to minimize this problem, we propose a template prediction method. A similar approach is introduced in Ref. 2, but our method is quite different in that every estimate required for the predictive template warping can be obtained from the feature estimators, so that the template prediction is maintained without additional framework for template state estimation. We assume that each feature template lies on a small planar surface in the 3-D scene space. Since we are not given any prior knowledge on the orientation of this surface, it is also assumed that the surface normal is parallel to

**Table 4** Comparison of MSE of camera parameters

| | $t_x$ | $t_y$ | $t_z$ | |
|---|---|---|---|---|
| PF | $2.83\times10^{-1}$ | $1.85\times10^{-1}$ | $3.33\times10^{-3}$ | |
| UKF | $1.06\times10^{-3}$ | $8.32\times10^{-2}$ | $1.62\times10^{-2}$ | |
| | $\mathbf{q}_0$ | $\mathbf{q}_1$ | $\mathbf{q}_2$ | $\mathbf{q}_3$ |
| PF | $5.12\times10^{-4}$ | $3.33\times10^{-3}$ | $1.56\times10^{-3}$ | $5.87\times10^{-2}$ |
| UKF | $6.12\times10^{-5}$ | $1.05\times10^{-4}$ | $7.85\times10^{-5}$ | $7.56\times10^{-6}$ |

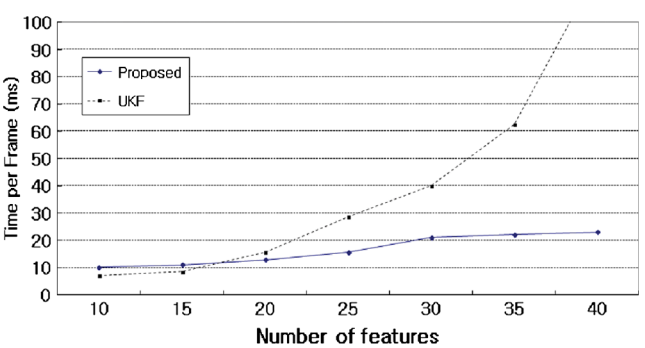


**Fig. 16** Processing time of our system and UKF-based tracking against number of features.

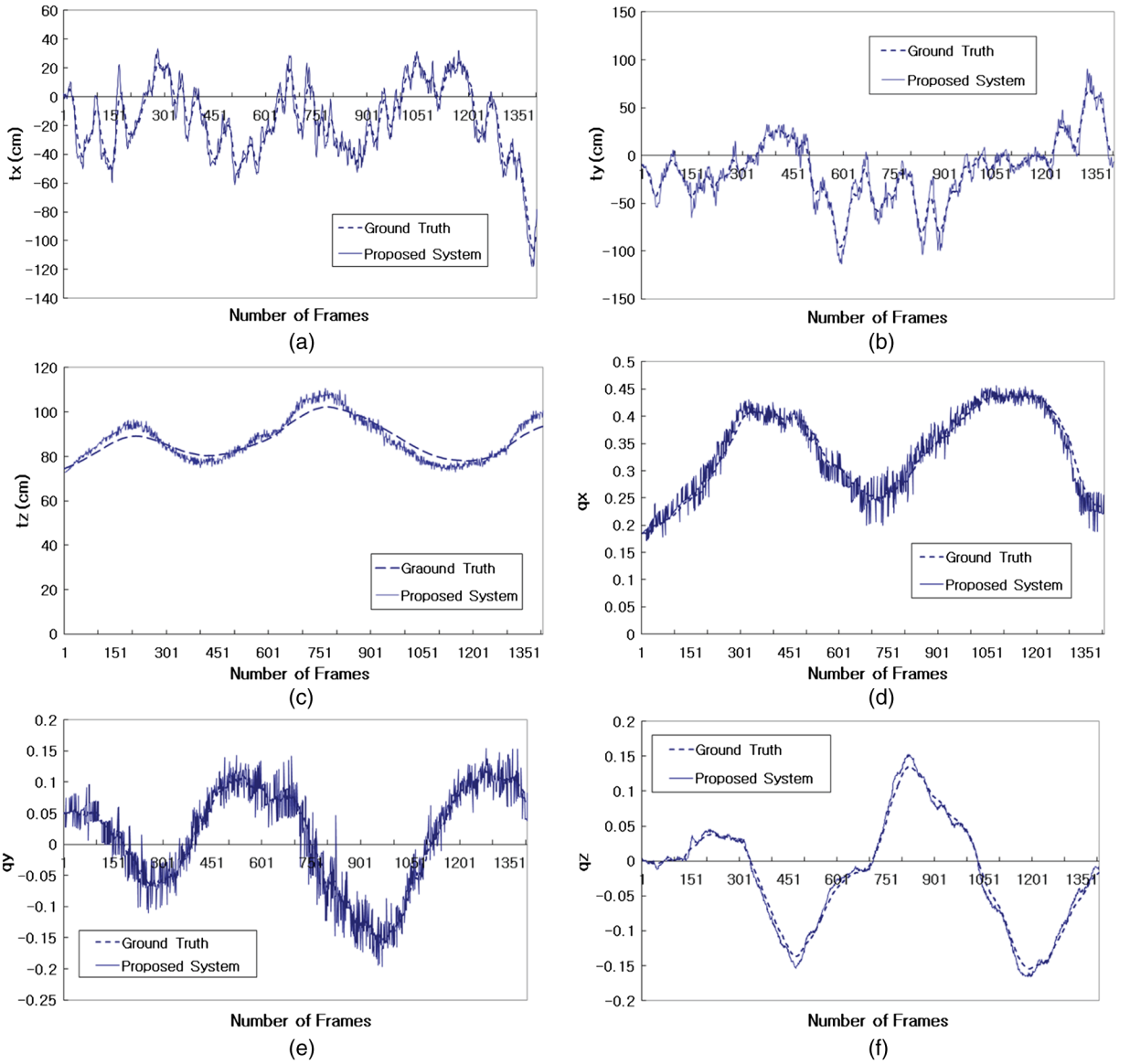


**Fig. 17** Tracking curves for comparing against ground truth. In each plot, the dashed line represents the true values of the camera parameters, while the solid line depicts those obtained by our algorithm for the 1400 frames of the sequences. (a) $t_x$, (b) $t_y$, (c) $t_z$, (d) $\mathbf{q}_1$, (e) $\mathbf{q}_2$, and (f) $\mathbf{q}_3$.

the projection ray that passes through the feature and the camera center at the initialization step. Once the 3-D coordinate and the depth are initialized by the scheme described in the previous section, each feature template is then stored in the form of a small planar patch. When a feature is tracked in the subsequent camera images, the feature template can be projected from the 3-D scene space to the image plane to produce a warped template for matching with the current image.

Suppose a template is located on a planar surface $\pi_k = (\mathbf{N}_k^{\mathrm{T}}, d_k)^{\mathrm{T}}$ whose surface normal is $\mathbf{N}_k^{\mathrm{T}}$ and depth to template is $d$; then two camera matrices are given by

$$\mathbf{P}_k = \mathbf{K}[\mathbf{I}|\mathbf{0}], \cdots \mathbf{P}_{k+1} = \mathbf{K}[\mathbf{R}|\mathbf{t}]. \tag{10}$$

Then, the projective transformation between the $k$'th image plane and that of the $k+1$'th time step in the reference frame is expressed by

$$\mathbf{H}_0 = \mathbf{R} - \frac{\mathbf{t}\mathbf{N}^{\mathrm{T}}}{d}. \tag{11}$$

From the projective mapping between the 3-D scene space and the image plane, the projective transformation $\mathbf{H}$ between the two image planes is computed as follows:[32]

$$\mathbf{H} = \mathbf{K}\left(\mathbf{R} - \frac{\mathbf{t}\mathbf{N}^{\mathrm{T}}}{d}\right)\mathbf{K}^{-1}. \tag{12}$$

This procedure is illustrated in Figs. 5 and 6, which depict an example of the predictive template warping. In the proposed framework, every estimate required for the template warping can be obtained from the feature trackers. The surface normal $\mathbf{N}_k$ of the plane $\pi_k$ is obtained directly from the feature tracker's projection ray $\bar{\mathbf{q}}_r$ of Eq. (5) and the scale parameter $d_k$ from the depth estimate. Therefore, the template prediction is maintained without additional framework

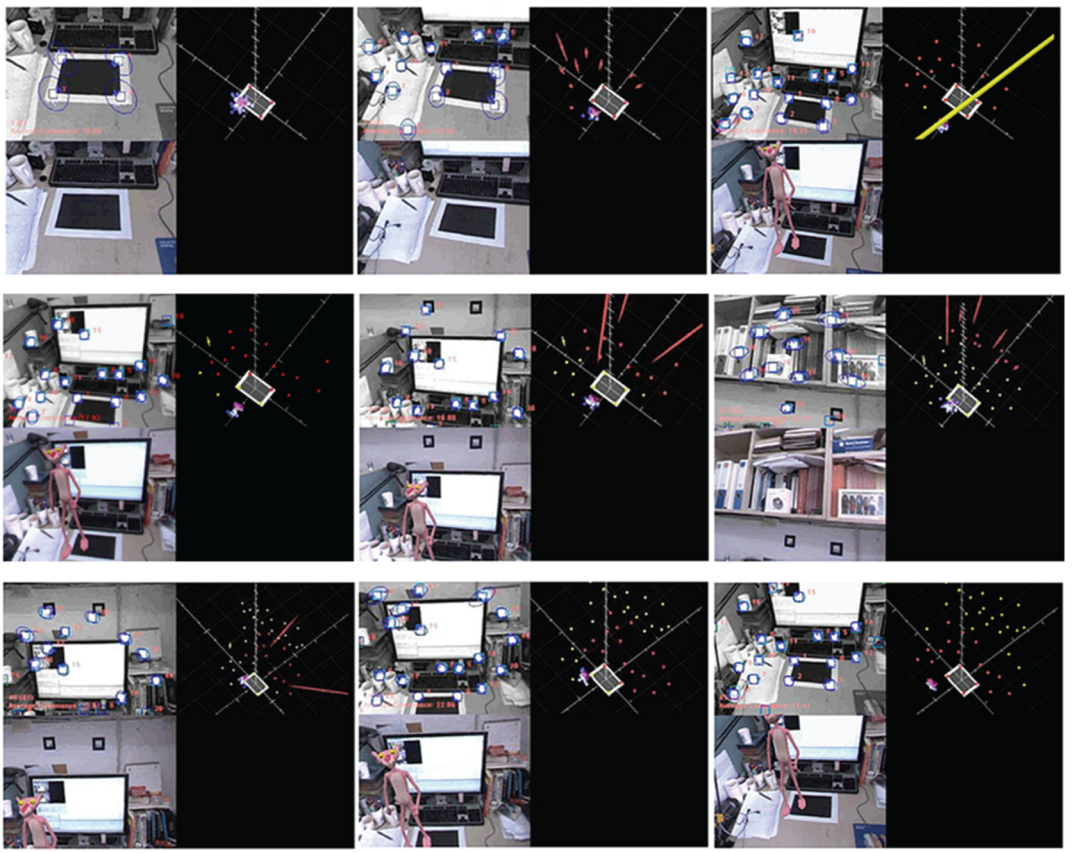

**Fig. 18** An example of the camera tracking in a desktop environment. A virtual object is augmented on the basis of geometric structure, which is obtained by the proposed system. (Video 3, QuickTime, 12.2 MB) [URL: http://dx.doi.org/10.1117/1.JEI.23.1.013029.3].

for template state estimation, and this may reduce the computational cost.

### 3.6 *Parallelization of Feature Mapping and Camera Tracking*

As the number of features increases, the computational cost of maintaining camera pose estimates rises rapidly, and this often prohibits real-time operation. In the proposed system, we split the camera tracking and feature mapping into two separate tasks, which are handled in two parallel processes. If the camera pose estimation and feature tracking can be separated independently, the camera tracking is less influenced by increase of the number of features, and therefore, more robust camera tracking and feature mapping can be achieved. This is illustrated in Fig. 7.

## 4 Results

Experiments are carried out using a desktop PC running MS Windows XP and a IEEE1394 camera with a resolution of $320 \times 240$ pixels at 30 frames per second. Figures 8(a) to 8(d) show that the camera tracking is achieved successfully over an extended period as the camera is moved with arbitrary motion over a desktop. The camera tracking is initialized with four known reference points in the scene corresponding to the corners of a rectangular pattern. In addition, Fig. 8 depicts the view through the camera and a 3-D view seen from the reference coordinate system, which also shows the camera trajectory. The views through the camera are augmented with the OpenGL teapot using the mean camera pose from the posterior, which also confirm that the tracking is quite stable. Also shown in Fig. 8 are the projections of the four reference points into each frame for each of the camera particles. The external view of the scene geometry represents the 3-D position of the particles and the mean camera state of the sample distribution. Figures 8(b) to 8(c) show tracking results for when the camera is being rapidly shaken and when the scene is severely occluded by an object. Note the abrupt spread of the particle distribution as camera shaking or occlusion occurs. This causes the filter to search wider over the state space in the active search phase until the feature templates are matched, while particles have equal weight. In the case that particles have uniform weight, camera samples cannot converge onto a correct camera pose. If this unstable state continues, the camera samples will uncontrollably spread out, and this may lead the tracking to a catastrophic failure. Figure 9 illustrates posterior distributions for each of the camera extrinsic parameters over the tracking sequence. In the figure, camera shake occurs between frames 110 and 165, and the severe occlusion between 165 and 180. In each case the estimated posterior spreads and then rapidly converges as shaking or occlusion stops. This verifies robustness of the camera tracking framework when it encounters unpredictable motion. Figure 10 shows processing time per period against number of particles and 15 features, and Fig. 11 illustrates the corresponding tracking accuracy.

In order to estimate the tracking error, we employed the RealViz® MatchMover package. It uses batch techniques

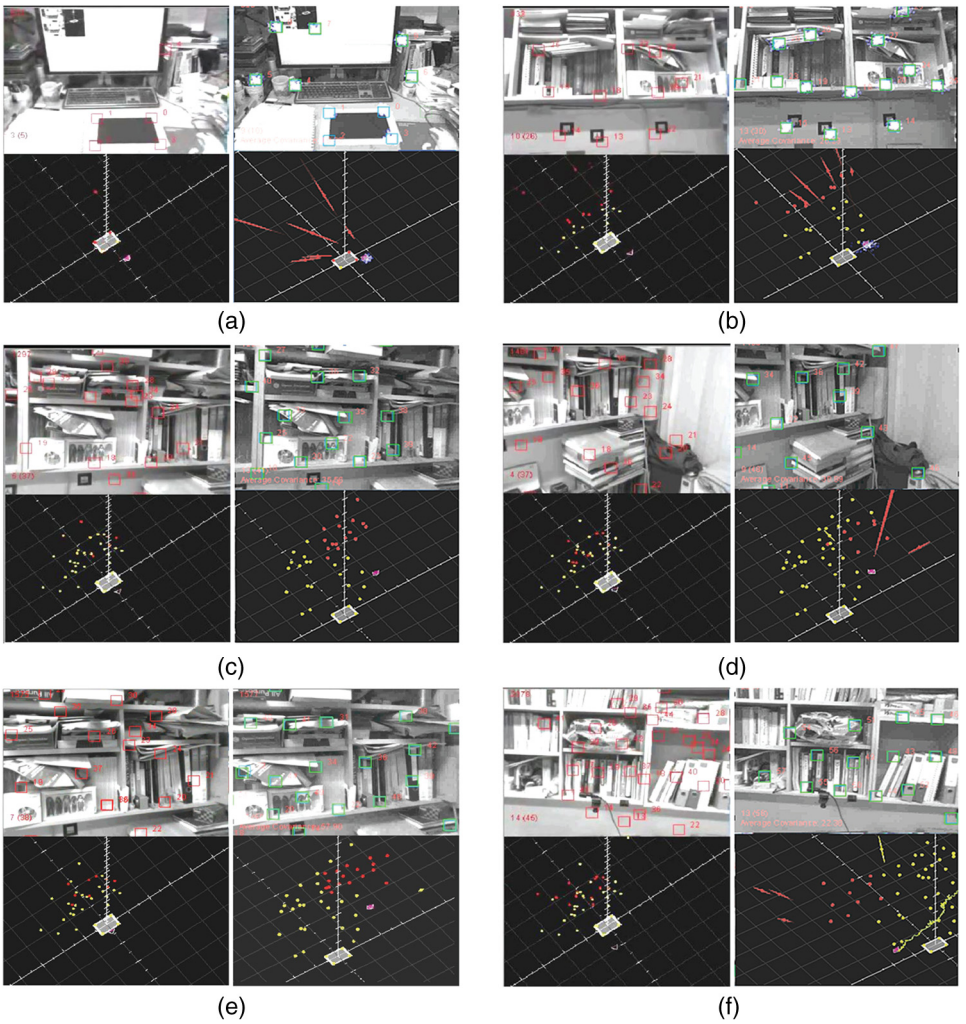


**Fig. 19** Camera tracking and feature mapping of UKF (left) and the proposed system (right). The UKF becomes unstable around frame 1000 and eventually fails to track after frame 1100 due to severe frame rate drop. (a) frame #530, (b) frame #383, (c) frame #1297, (d) frame #1475, (e) frame #1575, (f) frame #2378. (Video 4, QuickTime, 9.0 MB) [URL: http://dx.doi.org/10.1117/1.JEI.23.1.013029.4].

that yield very accurate results and allows manual intervention, which we have used to ensure correctness. We use its output as the ground truth of a series of sample image sequences. Then we compute the tracking data of the proposed system on the same sample image stream in order to compare the tracking accuracy. As the number of particles increases, processing timing also grows and tracking performance tends to be more accurate. Based on these results, we use 350 particles. In order to illustrate the benefits of the proposed framework, we compare its performance with that of the UKF as shown in Figs. 12 and 13.

Figures 12 and 13 show that the UKF becomes unstable and fails to track around frame 1100 due to rapid shake. On the other hand, in case of our system, the posteriors rapidly converge to the most probable particle, and this ensures that our system recovers within 3 to 5 frames and continues stable tracking even after the unpredictable motion.

Table 1 shows the performance of our system and those of other three camera tracking approaches. This table represents registration errors and time consumption to achieve camera pose estimation of each technique. The measurements are made for seven feature points whose locations are predefined. We see that UKF tracking is slightly faster and more accurate than the other systems. But the difference is quite trivial and our system also presents very good performance. In fact, time limit required for real-time operation is ~33 ms, and the difference between the systems is quite marginal. The registration errors are also presented. This error represents the distances between the predefined 3-D points and reprojected points by the tracking systems. If camera tracking is accurate, the reprojected points by tracking system will be coincident perfectly with the known points, and the distance will be close to zero. We can verify that the registration error is also acceptable regarding augmented reality application. Figure 14 shows an example of the feature tracking near the reference pattern. The system is initialized with four corner points of a black rectangular pattern. Each feature point is represented by an ellipsoid, which is parameterized by the feature covariance. Once a new feature is detected, an independent feature estimator is assigned to it. A new feature has very large uncertainty at the beginning of the tracking, but it is reduced quickly as the feature point converges. This is also verified from Fig. 15, which shows the resulting depth estimates of the feature point estimates. We can see large spikes in the depth estimates as new feature tracking filters are initialized, which then reduce rapidly as the estimates converge. Table 2

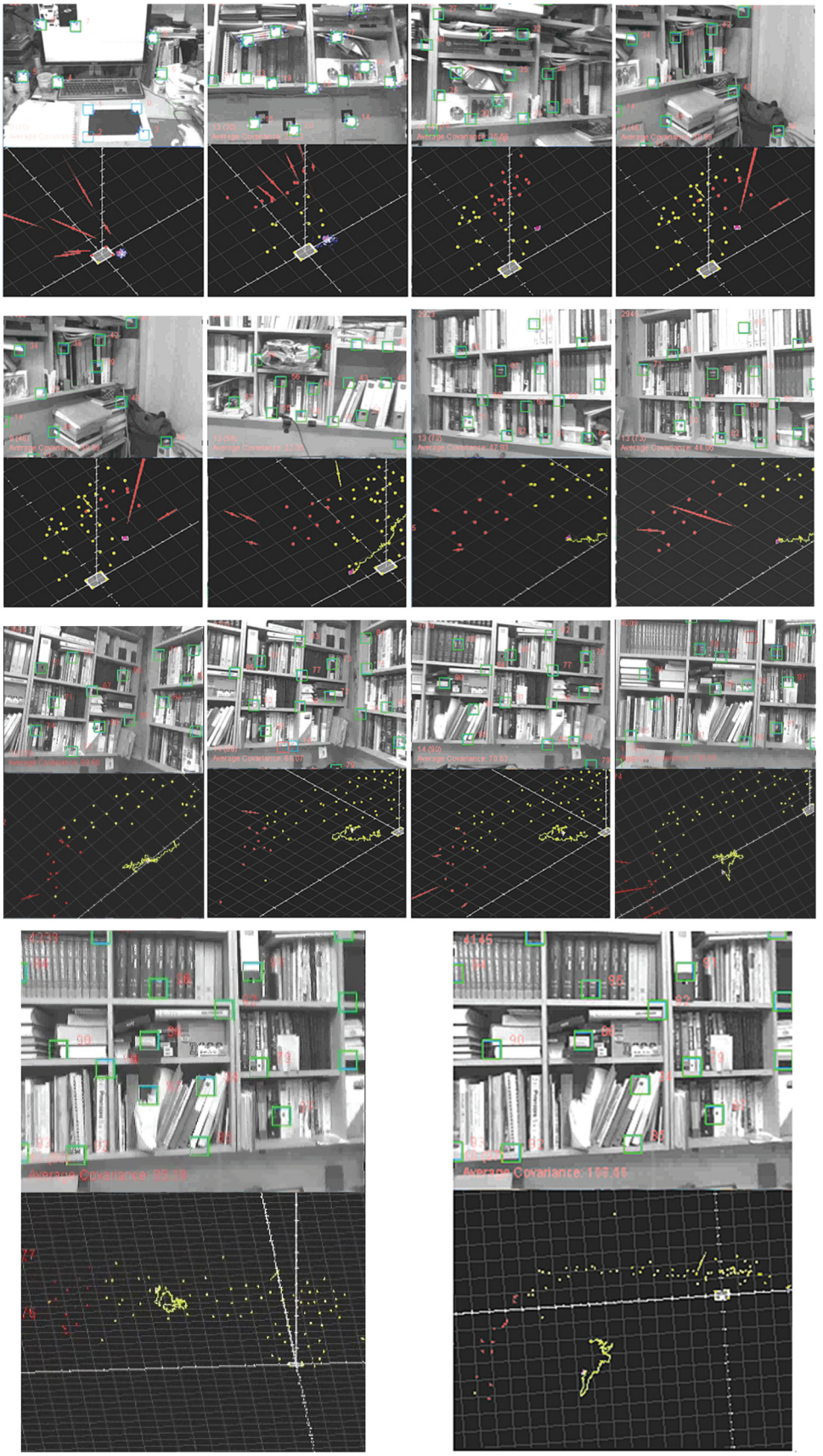

**Fig. 20** Real-time camera tracking and feature mapping. The camera starts at a place where the reference pattern is located and then moves away from the initialization point on the desktop. (Video 5, QuickTime, 17.4 MB) [URL: http://dx.doi.org/10.1117/1.JEI.23.1.013029.5].

shows that the proposed system operates in real time with a frame rate over ~24 frames per second when ~100 features are being mapped and tracked. All measurements are made using a PC, which has an Intel Core 2 Duo 2.66 GHz processor, 4 GB RAM, and an IEEE1394 camera with a resolution of $320 \times 240$ pixels.

In order to compare accuracy of our system against that of the UKF, we perform simulations based on synthetic geometric data. For the simulations, we first capture motion of the camera. That is, we record the estimated motion data during real-time camera tracking, and this recorded motion is used to generate synthetic measurement data. This approach makes it simple to perform simulations with a variety of realistic motions. Then, we generate noiseless image measurements of 10 co-planar reference points and obtain 3-D coordinates of the 10 reference points from UKF and the proposed framework, respectively. In order to compare the results, we compute differences between the results of the proposed method and those of UKF. In the simulation, our system employs a constant position motion model with Gaussian uncertainty, and the UKF is designed to use the same constant position model and all noise variances are set to be equivalent for both filters. Results are given in Tables 3 and 4. We see that UKF-based tracking presents slightly more accurate performance than the proposed system, but the difference is trivial. Therefore, we can conclude that the difference between the two tracking frameworks is not significant.

Time consumption per frame of the proposed and the UKF-based framework is presented in Fig. 16, which shows that the processing time of the conventional framework using a monolithic UKF grows as the number of features increases, and the computational cost exceeds the maximum capacity for ~30 feature points. Figure 17 illustrates tracking curves for comparing the accuracy of the proposed system against ground truth. The ground truth was generated with the MatchMover package, and the tracking data were computed from the same sample image stream. The tracking curves of the commercial software and the results of our method remain close to each other, and this shows that our system can perform robust tracking without critical tracking error.

Figure 18 illustrates an example that shows that the proposed system successfully tracks the camera moving over a desk environment. The upper-right images show the 3-D localization and mapping and the upper-left images show the projection of the mean and covariances of the mapped scene points in the camera view. The augmented scene is presented in the lower-left images.

We have compared the performance of the proposed framework with that given by a monolithic (full covariance) UKF-based system implemented in the same video sequence. Figure 19 shows the results of the camera tracking and feature mapping by the UKF and the proposed system. In each figure, the left images present the tracking results of the UKF framework, while the right column illustrates the camera is being tracked by our system. At the beginning of the test, both systems start to track the camera and then detect new features to track in real time. However, the UKF becomes quite unstable around frame 1000 and eventually fails to track after frame 1100 due to severe frame rate drop. In fact, this is also verified from Fig. 16, which shows that the processing time of the UKF rapidly grows as the number of features increases, and the computational cost starts to exceed the capacity for more than ~30 features (that is, around frame 1000 of the test video sequence). Figure 20 presents an example of the system traversing a room. The camera starts at a place where the reference pattern is located and then moves away from the initialization point on the desktop. More than 100 features are incorporated in the feature map, and 250 particles are used for the camera state estimation. The system runs at a frame rate of more than 24 Hz throughout the sequence. The results verify that the system also achieves successful camera tracking in a larger environment.

## 5 Conclusion and Future Work

We have presented a real-time camera tracking framework that combines a camera tracking filter and multiple feature estimators. The camera tracking is performed by a particle filter, while the feature tracking and mapping is achieved by independent feature trackers that perform unscented Kalman filtering. The camera tracking and feature mapping are split into two independent frameworks, which are treated in two parallel processes. In addition, we employed a template prediction technique in order to compensate change of the template appearance caused by the camera motion. The results show that the system achieves successful camera tracking and feature mapping in an AR environment. As a future work, we will conduct further study to employ a system that combines the FastSLAM framework and UKF for more robust camera tracking and feature mapping. We believe that the proposed framework can be employed as a useful tool in augmented reality systems and other camera-tracking-related works.

### References


1. A. Davison and N. Kita, "3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'01)*, pp. 384–391, IEEE, Kauai, Hawaii (2001).
2. A. Davison et al., "MonoSLAM: real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(6), 1052–1067 (2007).
3. A. Davison, "Simultaneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(7), 865–880 (2002).
4. B. Williams, G. Klein, and I. Reid, "Real-time SLAM relocalisation," in *Proc. IEEE Int. Conf. on Computer Vision*, pp. 1–8, IEEE, Rio de Janeiro, Brazil (2007).
5. M. Pupilli and A. Calway, "Real-time camera tracking using a particle filter," in *Proc. British Machine Vision Conf.*, pp. 43–57, BMVA, Oxford, UK (2005).
6. E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pp. 469–476, IEEE, New York, New Jersey (2006).
7. E. Eade and T. Drummond, "Monocular SLAM as a graph of coalesced observations," in *Proc. IEEE Int. Conf. on Computer Vision*, pp. 1–8, IEEE, Rio de Janeiro, Brazil (2007).
8. D. Chekhlov et al., "Robust real-time visual SLAM using scale prediction and exemplar based feature description," in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pp. 1–7, IEEE, Anchorage, Alaska (2007).
9. D. Chekhlov et al., "Real-time and robust monocular SLAM using predictive multi-resolution descriptors," in *Proc. Int. Symp. on Visual Computing*, Vol. 4292, pp. 276–285 (2006).
10. Y. Genc et al., "Marker-less tracking for AR: a learning-based approach," in *Proc. IEEE and ACM Int. Symp. on Mixed and Augmented Reality*, pp. 295–304, IEEE and ACM, Darmstadt, Germany (2002).
11. R. Subbarao, P. Meer, and Y. Genc, "A balanced approach to 3D tracking from image streams," in *Proc. 4th IEEE and ACM Int. Symp. on Mixed and Augmented Reality*, pp. 70–78, IEEE and ACM, Vienna, Austria (2005).

12. K. Xu, K. W. Chia, and A. D. Cheok, "Real-time camera tracking for marker-less and unprepared augmented reality environments," *Image Vis. Comput.* **26**(5), 673–689 (2008).
13. M. L. Yuan, S. K. Ong, and A. Y. C. Nee, "Registration using natural features for augmented reality systems," *IEEE Trans. Vis. Comput. Graph.* **12**(4), 569–580 (2006).
14. F. Ababsa and M. Mallem, "Robust camera pose tracking for augmented reality using particle filtering framework," *Mach. Vision Appl.* **22**(1), 181–195 (2011).
15. M. Hess-Flores, M. Duchaineau, and K. Joy, "Sequential reconstruction segment-wise feature track and structure updating based on parallax paths," in *The 11th Asian Conference on Computer Vision (ACCV'11), Lecture Notes in Computer Science*, Vol. 7726, pp. 636–649 (2011).
16. R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: dense tracking and mapping in real-time," in *Proc. IEEE Int. Conf. on Computer Vision*, pp. 2320–2327, IEEE, Barcelona, Spain (2011).
17. R. Castle, G. Klein, and D. Murray, "Wide-area augmented reality using camera tracking and mapping in multiple regions," *Comput. Vis. Image Underst.* **115**(6), 854–867 (2011).
18. G. Klein and T. Drummond, "Tightly integrated sensor fusion for robust visual tracking," in *Proc. British Machine Vision Conf.*, pp. 787–796, BMVA, Cardiff, UK (2002).
19. L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3D camera tracking," in *Proc. IEEE and ACM Int. Symp. on Mixed and Augmented Reality*, pp. 48–57, IEEE and ACM, Arlington, Virginia (2004).
20. R. A. Newcombe et al., "KinectFusion: real-time dense surface mapping and tracking," in *Proc. of the 2011 10th IEEE Int. Symp. on Mixed and Augmented Reality*, pp. 127–136, IEEE, Washington, DC (2011).
21. E. Bylow et al., "Real-time camera tracking and 3D reconstruction using signed distance functions," in *Proc. of Robotics: Science and Systems*, Berlin, Germany (2013).
22. T. Whelan et al., "Robust real-time visual odometry for dense RGB-D mapping," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 5724–5731, IEEE, Karlsruhe (2013).
23. J. Martinez-Carranza and W. Mayol-Cuevas, "Real-time continuous 6D relocalisation for depth cameras," in *Workshop on Multi View Geometry in Robotics (MVIGRO'13)*, Berlin, Germany (2013).
24. P. Henry et al., "RGB-D mapping: using depth cameras for dense 3D modeling of indoor environments," in *Proc. of the 12th Int. Symp. on Experimental Robotics*, pp. 22–25, ISER, New Delhi, India (2010).
25. G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE and ACM Int. Symp. on Mixed and Augmented Reality*, pp. 225–234, IEEE and ACM, Nara, Japan (2007).
26. D. Simon, *Optimal State Estimation*, John Wiley Sons, Hoboken, New Jersey (2001).
27. S. Soatto and P. Perona, "Reducing structure from motion: a general framework for dynamic vision. Part 1: Modeling," *IEEE Trans. Pattern Anal. Mach. Intell.* **20**(9), 933–942 (1998).
28. S. Soatto and P. Perona, "Reducing structure from motion: a general framework for dynamic vision. Part 2: implementation and experimental assessment," *IEEE Trans. Pattern Anal. Mach. Intell.* **20**(9), 943–960 (1998).
29. E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Proc. IEEE Int. Conf. on Computer Vision*, pp. 1508–1511, IEEE, Beijing, China (2005).
30. E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. European Conf. on Computer Vision*, Lecture Notes in Computer Science, Vol. 3951, pp. 430–443, Graz, Austria (2006).
31. J. Civera and A. Davison, "Inverse depth to depth conversion for monocular SLAM," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2778–2783, IEEE, Rome, Italy (2007).
32. R. Hartely and A. Zisserman, *Multiple-View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, Cambridge, United Kingdom (2003).

**Seok-Han Lee** received his BS degree in electronic engineering from Chung-Ang University in 1999 and his MS and PhD degrees in image engineering from Chung-Ang University in 2001 and 2009, respectively. From 2001 to 2004, he worked as an engineer in LG Electronics. He is currently working as a professor in the Department of Information and Communication Engineering of Jeonju University. His research interests include computer vision and three-dimensional reconstruction.