Real-Time Camera Tracking Using a Particle Filter

Mark Pupilli and Andrew Calway

Department of Computer Science
University of Bristol, UK

{pupilli,andrew}@cs.bris.ac.uk

Abstract

We describe a particle filtering method for vision based tracking of a hand held calibrated camera in real-time. The ability of the particle filter to deal with non-linearities and non-Gaussian statistics suggests the potential to provide improved robustness over existing approaches, such as those based on the Kalman filter. In our approach, the particle filter provides recursive approximations to the posterior density for the 3-D motion parameters. The measurements are inlier/outlier counts of likely correspondence matches for a set of salient points in the scene. The algorithm is simple to implement and we present results illustrating good tracking performance using a 'live' camera. We also demonstrate the potential robustness of the method, including the ability to recover from loss of track and to deal with severe occlusion.

1 Introduction

Recent years have seen the emergence of vision based algorithms aimed at tracking the location and pose of a camera in real-time. This is a challenging task, requiring mechanisms beyond that previously explored in traditional 'structure from motion' (SFM), particularly if the camera is undergoing general motion such as that exhibited by a hand held or wearable camera. The majority of methods have therefore relied on off-line pre-calibration of 3-D scene structure to stabilise tracking [11, 5, 7]. Recently, however, significant advances have been made in simultaneously estimating structure whilst tracking, opening up the possibility of more flexible general purpose algorithms. The most notable is the system developed by Davison [3], in which a Kalman filter is used for sequential estimation of the motion parameters, based on a calibrated camera and judicious maintenance of strongly salient features in the scene. This system has demonstrated impressive tracking performance, particularly when using a wide angled lens [4]. Other recent methods include the real-time implementation of RANSAC based structure from motion described by Nistér [12] and the Kalman filter approach of Chiuso et al [2].

In this paper, we build on this work and describe a new approach to real-time camera tracking which is similar in concept to that in [3], but based within a particle filtering framework. The Kalman filter used in [3] works well if approximate linearity is maintained between time steps and if the estimates and measurements are consistently 'good' in a unimodal Gaussian sense. However, it is well known that deviation from this can

BMVC 2005 doi:10.5244/C.19.50

cause instability and ultimately tracking failure. Effects such as ambiguity in feature location, occlusion of features and erratic motion jitter may cause problems, and recovery from loss of track is likely to be difficult. This is addressed in [3] by ensuring that only measurements from strongly salient features are used to update the filter. However, such constraints may not always be possible or desirable, and in any case it does not necessarily address the problem of recovery. This suggests that moving towards a more general Bayesian sampling framework, such as that provided by the particle filter, would be beneficial, particularly if the methods are to be extended to non-static scenes in which non-linearities and multi-modality are likely to be more significant. Particle filters deal with these in a natural way and as such could provide improved robustness over the Kalman approach. They also tend to be more flexible, particularly with respect to the observation model, and in general they are simpler to implement.

Bayesian sampling has been used previously for vision based SFM. Forsyth et al [6] investigated the use of general Monte Carlo sampling for SFM, whilst Qian and Chellappa describe a method based on sequential importance sampling (SIS) [13]. Their approach uses partitioning to reduce the dimensions of the sampling space, separating the estimation of motion from that of depth using the epipolar constraint. However, both this method and that of [6] are presented in the context of traditional off-line SFM and real-time operation is not discussed. The closest to a real-time method is that described by Chang and Hebert [1], who use a particle filter for robot localisation based on an omnidirectional camera. This is a bottom-up approach in which the filter is used for time integration of two-frame SFM estimates, combined with sampling over feature correspondences. However, again no details of real-time operation are given.

We adopt a top-down state space approach, similar to that in [13], but sampling only on the motion parameters. The measurements are based on inlier/outlier counts of correspondence matches for a set of estimated 3-D scene points. These are obtained dynamically during tracking using an auxiliary process, boot-strapped by a set of known 3-D points at initialisation, in a similar manner to that used in [3]. This removal of depth estimation from the main particle filter is the key to real-time operation. We use SIS with resampling, i.e. the Condensation algorithm [9], and use annealed sampling [10] at each iteration to improve stability. The algorithm is simple to implement and computation times are linear in the number of particles and scene points. An overview of the algorithm is given in the next section. This is followed by details of the main components in Sections 3-5 and results illustrating real-time tracking for several different scenarios are presented in Section 6.

2 Algorithm Overview

The basis of the method is to use the particle filter to provide sequential approximations to the posterior density for the 3-D motion parameters as the camera moves. The effective number of state dimensions is therefore 6, representing the 3-D location and rotation of the camera with respect to a world coordinate system. We use a calibrated camera, and so the motion estimates are metric, and the coordinate system is aligned with the initial camera coordinate frame. Each particle therefore corresponds to a potential 3-D location and pose as illustrated in Fig. 1a, which shows 'camera particles' at time k, each defined by a translation $\mathbf{t}_k^n$ and rotation R_k^n, where the circles represent the particle weights. These

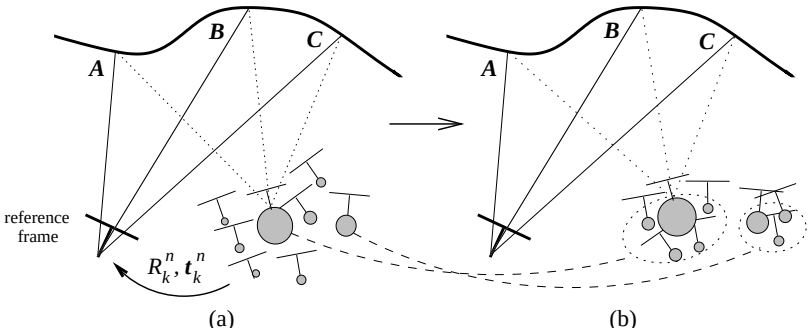


Figure 1: Camera tracking using a particle filter. Each weighted particle represents a potential 3-D location and pose for the camera, and the weights (indicated by circles) give successive approximations to the posterior density for the motion parameters.

give the approximation to the posterior density and are used to proportionally distribute new particles in the next time frame using Condensation [9], with particles concentrated around potential modes in the density as illustrated in Fig. 1b.

As with all particle filters, the key components are the state dynamics and the observations used. Without prior knowledge of camera movement, we use a simple random walk for the change in location and pose, although alternatives such as a constant velocity model could also be used within the same framework. More important is the way in which we compute the particle weights at each time step, i.e. the likelihood of the observations for a given state. This is best described by first assuming that we have a known set of 3-D points in the scene, such as the points A-C in Fig. 1. The likelihood for given particle is then based on projecting the 3-D points onto the associated 'camera' and comparing the neighbourhoods in the current frame about the projected points with those in the reference frame. In other words, we match local templates according to the relative motion $(R_k^n, \mathbf{t}_k^n)$ of the particle. In fact, we show later that we can base the likelihood on a simple inlier/outlier count amongst the scene points.

We adopt a strategy similar to that in [3] for obtaining the 3-D scene points. The system is boot-strapped by a set of known 3-D points which enables an initial particle distribution to be built up. As tracking progresses, new 3-D points are introduced by identifying salient points in the frames and estimating their depths by sequential reprojection and triangulation of the camera particles (in this respect it differs from the method used in [3]). On convergence of a depth estimate, a new point is incorporated into the scene map and then used in computing the likelihood. This enables tracking to continue as the field of view moves away from the initialisation points. Details of this and the other components of the algorithm are given in the following sections.

3 State and Observation Models

We denote the motion state at time k by the vector $\mathbf{x}_k$, which defines the 3-D location and pose in terms of the translation vector $\mathbf{t}_k$ and rotation matrix R_k, respectively. For computational reasons we use a quarternion to represent the rotation, giving the state vector $\mathbf{x}_k = (\mathbf{q}_k, \mathbf{t}_k)$, where the quarternion is normalised so that $(q_{ks}^2 + q_{kx}^2 + q_{ky}^2 + q_{kz}^2) = 1$.

Given frame k, we take measurements to give observations $\mathbf{y}_k$ and the set of observations up to and including time k are denoted $\mathbf{y}_{1:k}$. We also assume that we have a set of 3-D scene points, $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$, which are defined in the reference coordinate system. To ease description, the number of points is assumed to be fixed; the removal of points or dynamic introduction of new points is easily accommodated. For a point $\mathbf{z}_i$, its projection onto the image plane of a camera with motion state $\mathbf{x}_k$ is then denoted $\mathbf{u}(\mathbf{z}_i, \mathbf{x}_k)$ and this is given by

$$\mathbf{u}(\mathbf{z}_i, \mathbf{x}_k) = \Pi(R_k \mathbf{z}_i + \mathbf{t}_k) \tag{1}$$

where Π is the non-linear projection operator assuming an ideal pin-hole camera. We use a calibrated camera and thus the parameters of Π are assumed to be known.

Tracking then amounts to obtaining successive approximations to the posterior density $p(\mathbf{x}_k|\mathbf{y}_{1:k}, Z)$. Condensation provides this in the form of sets of weighted particles $\{(\mathbf{x}_k^1, w_k^1), \ldots, (\mathbf{x}_k^N, w_k^N)\}$, where the particle $\mathbf{x}_k^n$ is a state space sample and the weights are proportional to $p(\mathbf{y}_k|\mathbf{x}_k^n)$, such that $\sum_{n=1}^N w_k^n = 1$. The algorithm requires a probabilistic model for the state evolution between time steps, ie $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, and a likelihood $p(\mathbf{y}_k|\mathbf{x}_k, Z)$ for the observations given the state and the structure [9]. We adopt a random walk for the former, based on a uniform density about the previous state

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathrm{U}(\mathbf{x}_{k-1} - \mathbf{v}, \mathbf{x}_{k-1} + \mathbf{v}) \tag{2}$$

where $\mathbf{v}$ represents our uncertainty about the incremental camera movement. We found that this gave better performance than the normal density, particularly in terms of responsiveness to small rapid movements. The model could also include a deterministic element, such as constant velocity, although we have not found this to be necessary.

We base the likelihood $p(\mathbf{y}_k|\mathbf{x}_k, Z)$ on the closeness of the projected points $\mathbf{u}(\mathbf{z}_i, \mathbf{x}_k)$ to neighbourhoods in frame k which have a high correlation with the templates in the reference frame. Denoting frame k by $I_k(\mathbf{u})$, we define a set of correlation fields, one for each scene point, as follows

$$\rho_{ki}(\mathbf{u}) = T_i(\mathbf{u}) * I_k(\mathbf{u}) \tag{3}$$

where $*$ denotes normalised cross correlation and $T_i(\mathbf{u})$ is the template, a small neighbourhood about the projection of $\mathbf{z}_i$ into the reference frame. The set of observations are then those points for which the correlation values are above a threshold, i.e. $\mathbf{y}_k = \{\mathbf{y}_{k1}, \ldots, \mathbf{y}_{kM}\}$, where the subset $\mathbf{y}_{ki}$ relates to $\mathbf{z}_i$ and its template $T_i(\mathbf{u})$, and is given by

$$\mathbf{y}_{ki} = \{\mathbf{u} : \rho_{ki}(\mathbf{u}) > \varepsilon_\rho\} \tag{4}$$

where ε_ρ is the correlation threshold. For the likelihood we then use a function related to the number of scene points whose projections into the frame are within a given radius of at least one relevant observation, i.e.

$$p(\mathbf{y}_k|\mathbf{x}_k, Z) \propto \exp(-\sum_{i=1}^{M} \prod_{\mathbf{u} \in \mathbf{y}_{ki}} d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k)) \tag{5}$$

where $d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k)$ indicates whether the point $\mathbf{z}_i$ is an inlier or outlier with respect to the observation at $\mathbf{u}$ and the state $\mathbf{x}_k$

$$d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k) = \begin{cases} 1 & \text{if } ||\mathbf{u} - \mathbf{u}(\mathbf{z}_i, \mathbf{x}_k)|| > \varepsilon_d \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and the threshold ε_d defines the 'tolerance' in the template matching. An example is illustrated in Fig. 2a, which shows thresholded correlation fields (shaded) for three templates A-C and projections of scene points for two camera particles (crosses and squares). The circles denote the threshold ε_d. In this case, the particle represented by the crosses gains support from all three points, whilst the other particle is only supported by point B. This form of likelihood has similarities with that used in RANSAC [8] and we found that it gave better performance than the alternative of using a continuous kernel.

Note also that the correlation values need not be computed over the whole frame - projections of particles propagated from the previous frame can be used to identify areas within which observations can contribute to particle weights. This enables the likelihoods to be computed efficiently. It also has the advantage that matches for points which become inactive, due to occlusion or feature mismatch, will continue to be sought along highly probable trajectories, giving the potential for recovery.

4 Particle Annealing

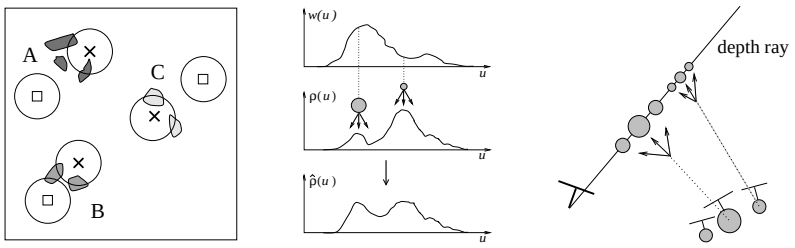
The above likelihood function has the advantage of being quick to evaluate. This is important for real-time operation since it needs to be computed for every particle at each time step. However, its discrete form means that care must be taken in setting the threshold ε_d. Setting it too high results in reduced discrimination which can cause drift. Reducing it increases discrimination, but requires greater coverage of the sampling space so that enough samples are drawn around significant modes. Unfortunately, increasing the number of samples to achieve this quickly becomes prohibitive. To address this, we use particle annealing [10], in which samples are iteratively focused onto potential modes, whilst gradually reducing the threshold ε_d.

At each time step, we use a large value for ε_d to obtain an initial set of weighted particles. Annealing then proceeds by resampling from this set using a uniform distribution with a smaller width than that used to propagate the samples from the previous time step ($\mathbf{v}$ in (2)). An updated set of weighted particles is then obtained using a smaller value of ε_d in the likelihood and this initialises the next annealing step. The process continues using smaller and smaller values of $\mathbf{v}$ and ε_d, resulting in greater concentration of the samples around modes in the state space. Further details on particle annealing are given in [10].

We also re-weight the correlation values used in the likelihood at each annealing step according to the current estimate of the motion posterior density. This further increases discrimination by penalising ambiguous matches. Re-weighting is similar to the calculation of the likelihood, but in reverse. Correlation values for scene point i are re-weighted so that $\rho_{ki}(\mathbf{u}) \rightarrow \alpha_{ki}(\mathbf{u})\rho_{ki}(\mathbf{u})$, where

$$\alpha_{ki}(\mathbf{u}) = \sum_{n=1}^{N} w_k^n (1 - d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k^n)) \tag{7}$$

and $d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k^n)$ indicates whether $\mathbf{z}_i$ is an inlier ($d() = 0$) or outlier ($d() = 1$) with respect to the point $\mathbf{u}$ and the particle $\mathbf{x}_k^n$ as defined in (6). In effect, the particle weights are projected down into the correlation field and diffused locally, boosting values in the vicinity of particle clusters with high probability. This increases support for significant modes and hence accelerates the annealing process. Note also that since the particle weights are derived from measurements based on all the known scene points, the re-weighting acts



(a) (b) (c)

Figure 2: (a) Inlier/outlier support for camera particles (crosses and squares); (b) correlation re-weighting; (c) depth estimation for new 3-D scene points.

as a 'pooling' mechanism, which helps to sort out feature ambiguities or mismatches. In practice, we base the re-weighting on a subset of particles, properly drawn from the current distribution, to minimise computational effort. A simple example of re-weighting is illustrated in Fig. 2b, which shows a 1-D correlation field before and after re-weighting using the projected particle distribution $w(u)$. In this case, re-weighting has the effect of boosting the weaker left-hand mode in the original correlation field.

5 Initialisation of 3-D Scene Points

We now consider how we obtain the 3-D points Z. One option is to build depth estimation into the particle filter as in [13]. However, this significantly increases the number of state dimensions and although partitioning can be used [13], we found this to be unreliable and also requiring a significant increase in the number particles, making it prohibitive for real-time operation. Instead, we follow the strategy adopted in [3], and use an auxiliary process to build points dynamically into a scene map, boot-strapped by a small number of known 3-D points at initialisation.

Since we use a calibrated camera, initialisation is straightforward. The camera is held parallel to a plane and points lying on the plane are selected by hand. This calibrates the system and tracking can proceed based on the initialised 3-D points. New points are introduced by detecting salient points in subsequent frames and then iteratively building up depth estimates. The process works as follows. The region around a point forms a new template and the point defines a 3-D ray with respect to the camera frame. Weights associated with discrete depths along the ray are then updated by measurements from subsequent frames to obtain a depth density. In [3], this is achieved by projecting each depth onto each frame and seeking support based on feature matching. Here we adopt the reverse process and, for a subset of properly sampled camera particles, triangulate the ray with points having high correlation with the new template (using least squares) and updating the depth weights in the vicinity of the triangulated point as illustrated in Fig. 2c. This avoids updating every depth sample and allows large depth ranges to be dealt with efficiently. The update is based on re-weighted correlation values, where the re-weighting is similar to that used in the annealing, except that the inlier/outlier count is based on a weighted distance measure from the epipolar line defined by the depth ray. This is important since it pools the measurements from active points already in the scene map. The process continues until either the weights converge on a given depth, in which case the new point is incorporated into the scene map, or the estimation is abandoned.

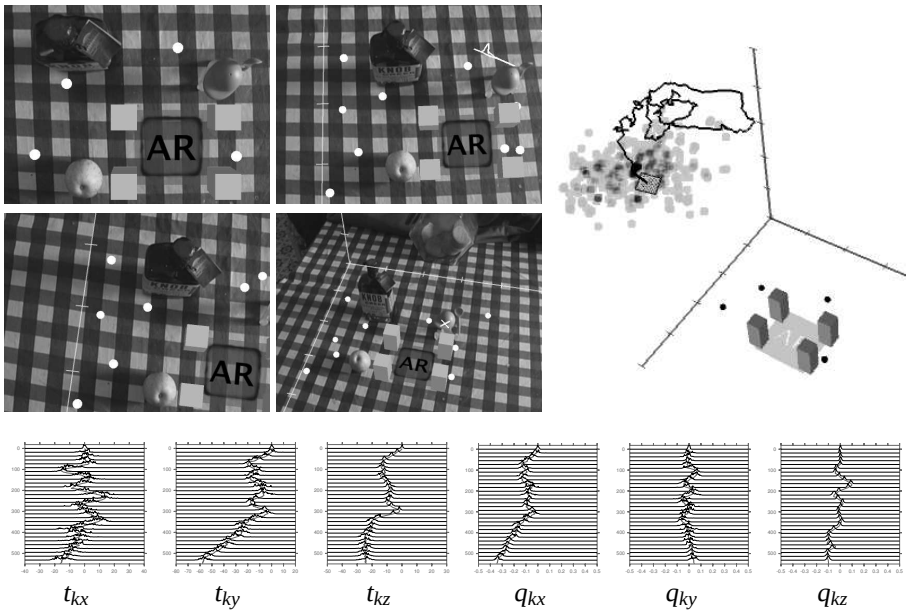


Figure 3: Camera tracking over a kitchen table with augmented graphics: (top-left) example frames with projections of scene points; (top-right) mean camera trajectory and particle weights; (bottom) projections of the posterior density for each motion parameter.

6 Results

Examples of real-time tracking are shown in Figures 3, 5 and 6. We used a hand held camera and the frames were processed using a 3GHz workstation. Figure 3 shows 22 seconds of tracking as the camera moves smoothly over a kitchen table. Tracking was initialised by 4 points on the table as shown in the top-left image. Note that tracking was successful despite the feature ambiguities caused by the checked tablecloth. Six new points were introduced and their projections are shown in the other images, along with the depth weights for one of the points projected down onto the epipolar line prior to convergence (top-right image; the projection of this new point is shown in the bottom-left image). The cross in the bottom-right image indicates that a point has become inactive due to low correlation values, here caused by occlusion. Tracking accuracy was assessed by augmenting the scene with graphics placed on the 3-D initialisation plane. For simplicity, the weighted mean of the particles was used for the camera motion. Although this is less than ideal in the presence of multiple modes, resulting in a small amount of jitter, the graphics are generally stable, indicating good tracking performance.

The 3-D plot in Fig. 3 shows the mean camera trajectory and the particle weights for the location in the current frame, where black indicates high weight. Note the presence of several modes in the distribution. The effects of annealing are illustrated in Fig. 4, which shows the particle distribution prior to (left) and after several steps of annealing (middle and right). Note how the particles are focused onto two separate modes. In practice, we have found that 2 or 3 steps of annealing at each time step is required to give stable tracking. The plots in Fig. 3 show the temporal evolution of the projected posterior

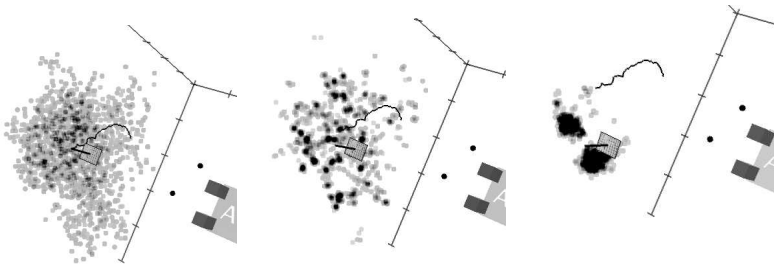


Figure 4: Particle annealing is used to focus samples around significant modes.

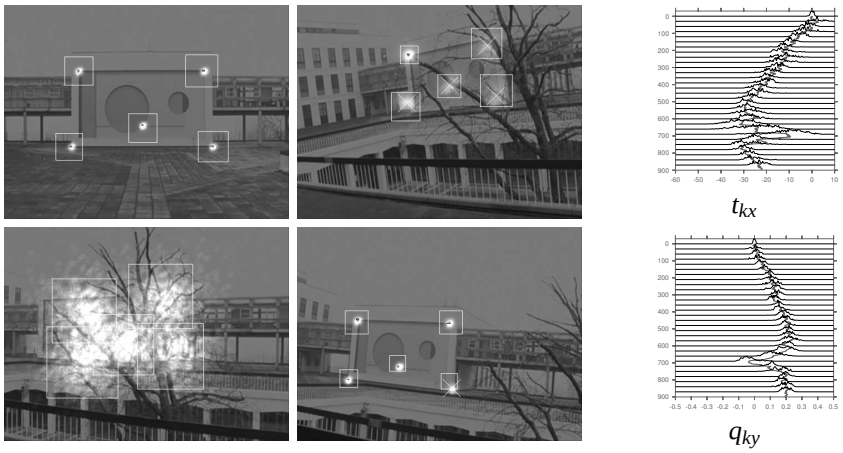


Figure 5: Outdoor tracking illustrating recovery from severe occlusion.

density for each of the motion parameters. The filter was using 500 particles and the processing rate was 40 fps whilst using 4 active points, dropping to 25 fps with 8 active points. At present the introduction of new points is not optimised and so processing time was reduced to 20 fps when introducing one point at a time. We anticipate that this can be significantly increased. Similar processing speeds apply to the other examples.

Figure 5 demonstrates successful tracking outdoors and robustness to severe occlusion. Five points on a planar sculpture were used for initialisation and the camera was moved in an unstable manner to the right whilst rotating to keep the sculpture in view. The images show the projections of the 3-D points for the set of camera particles, creating a 'cloud' around each salient feature. The bounding box for computing the correlation values is also shown. Part way through tracking, the sculpture is occluded by a tree, causing temporary loss of track. Note the dispersion of the particle clouds in the top-right image as occlusion begins and the wide spread of particles once occlusion has occurred in the bottom-left image. At this point the filter has lost track but recovers in a controlled way by wide distribution of motion particles across the state space. This can be clearly seen in the projections of the posterior density shown on the right (occlusion begins around frame 600). Note the sudden widening of the density as occlusion occurs and the narrowing as the filter manages to lock onto the features once occlusion has passed and tracking is successfully continued as shown in the bottom-right image.

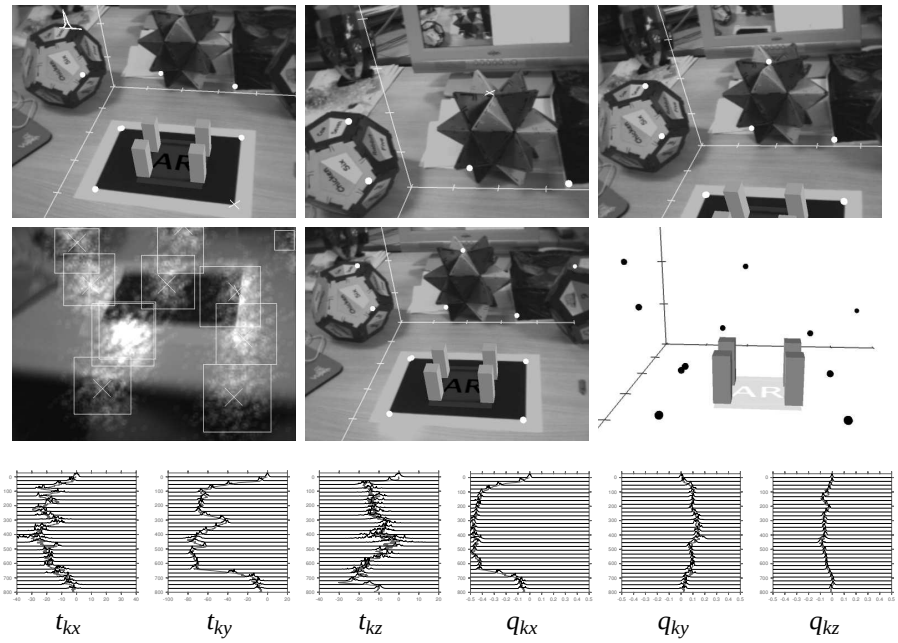


Figure 6: Tracking away from the initialised structure and recovery from camera shake.

The example in Fig. 6 illustrates how the introduction of new scene points allows tracking to continue away from the initialisation points. Four points are initialised on a desk using a calibration pattern. As the camera moves away, new points are incorporated as shown in the top-left image and tracking continues as the initial points move out of the field of view. Eventually tracking is based purely on newly incorporated points as shown in the top-middle image. When the camera returns to its original position the augmented graphics appear stable and in their correct position, indicating successful estimation of the camera trajectory. The bottom-right figure shows the estimated 3-D points. This example also illustrates recovery following severe camera shake. Part way through tracking (around frame 400) the camera was shaken whilst moving. As tracking is lost, the camera particles disperse as shown by the projected clouds in the bottom-left image and in the projected posterior density. However, once shaking ceases, the filter quickly recovers, locking on to the scene points and tracking continues successfully.

7 Conclusions

We have demonstrated that real-time camera tracking can be achieved using a particle filter, with minimal pre-calibration of structure and the ability to incorporate new structure during tracking. This gives the potential for general purpose camera tracking. It shares many of its central concepts with the system described in [3], but we believe that the use of the particle filtering framework opens up the possibility of greater robustness, particularly in terms of recovery as illustrated in the results and the potential to deal with the increased multi-modalities likely to be present when extending the idea to non-static scenes. The simplicity of the particle filter and the flexibility in defining the observation model are also

key advantages. As well as extending the technique to non-static scenes, the mechanism for introducing new points needs to be optimised; our current implementation is probably not as robust as that developed in [3], primarily due to the fact that we are not currently incorporating uncertainty about the camera position when initialising new points. Work on this and extensions of the method to allow wide area tracking is in progress.

Acknowledgement This work was funded by the EPSRC Equator IRC.

References

[1] P. Chang and M. Hebert. Robust tracking and structure from motion through sampling based uncertainty representation. In *Proc IEEE Int Conf on Robotics and Automation*, 2002.

[2] A Chiuso, P Favaro, H Jin, and S Soatto. Structure from motion causally integrated over time. *IEEE Trans on Patt Analysis and Machine Intell*, 24(4):523–535, 2002.

[3] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. International Conference on Computer Vision*, 2003.

[4] A.J. Davison, Y. González Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon*, 2004.

[5] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans Patt Analysis and Machine Intell*, 24(7):932–946, 2002.

[6] D.A. Forsyth, J. Haddon, and S. Ioffe. The joy of sampling. *Int Journal of Computer Vision*, 41(1/2):109–134, 2001.

[7] I. Gordon and D.G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *Int Symp on Mixed and Augmented Reality*, 2004.

[8] R Hartley and A Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int Journal of Computer Vision*, 29(1):5–28, 1998.

[10] A. Blake J. Deutscher and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proc Int Conf Computer Vision and Pattern Recognition*, 2000.

[11] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *Proc ACM Symp on Virtual Reality Software and Technology*, 1997.

[12] D. Nistér. Preemptive ransac for live structure and motion estimation. In *Proc. International Conference on Computer Vision*, 2003.

[13] G. Qian and R. Chellappa. Structure from motion using sequential monte carlo methods. *Int Journal of Computer Vision*, 59:5–31, 2004.