Extended Keyframe Detection with Stable Tracking for Multiple 3D Object Tracking

Youngmin Park, *Student Member*, *IEEE*, Vincent Lepetit, and Woontack Woo, *Member*, *IEEE*

Abstract—We present a method that is able to track several 3D objects simultaneously, robustly, and accurately in real time. While many applications need to consider more than one object in practice, the existing methods for single object tracking do not scale well with the number of objects, and a proper way to deal with several objects is required. Our method combines object detection and tracking: frame-to-frame tracking is less computationally demanding but is prone to fail, while detection is more robust but slower. We show how to combine them to take the advantages of the two approaches and demonstrate our method on several real sequences.

Index Terms—Augmented reality, computer vision, object tracking, object detection.

1 Introduction

Single 3D object tracking using natural features has now been well explored by the Computer Vision and Augmented Reality communities [1]. By contrast, the multiple object case, where several 3D objects must be considered simultaneously, has not really been addressed, while it is often needed for real applications, for example, a tangible interface.

In this paper, we apply image feature recognition and keyframe techniques to several objects because they have been shown to be very robust for the single object case [2] and also for SLAM [3] by preventing loss of track and drift. Unfortunately, it is not possible to directly apply the method described in [2] in practice because its complexity grows with the number of objects.

We therefore propose an approach that is able to efficiently deal with several objects without concession to robustness. In addition, it is particularly stable. The key is an efficient combination of feature recognition and feature tracking from frame-to-frame. Because feature recognition is usually very time consuming, it is indeed very tempting to introduce feature tracking, which is generally cheaper. However, this has to be done carefully to avoid error propagation that eventually leads to drift.

Our approach is somehow related to [4] but is much more robust since ours exploits feature recognition. For each target object, we have its CAD 3D model plus a small set of registered reference images we will call "keyframes" of the object. Some example keyframes are shown Fig. 2. We try to match each input frame against only a subset of the keyframes to keep processing at frame rate, and track features lying on the visible objects over consecutive frames. The two sets of matches, those with the previous frame and those with the keyframes, are correctly combined by propagating errors to estimate the object 3D poses. Thus, robustness is achieved by regularly "injecting" features from the keyframes in the computation. Moreover, tracking features from frame-to-frame gives stability and relaxes the need for considering every keyframe at every frame, keeping the method efficient. We also take into account the object visibility to properly handle occlusion between the tracked objects. A preliminary version of this work was published in [5]. This paper describes several technical improvements, presents more experiments, and discusses the recent literature.

In the remainder of the paper, we first discuss related work on single and multiple object tracking. Then, we describe our method and present our results.

- *Y. Park and W. Woo are with the U-VR Lab., Department of Information and Communications, Gwangju Institute of Science and Technology, Oryong-dong, Buk-gu, Gwangju, South Korea. E-mail: {ypark, wwoo}@gist.ac.kr.*
- *V. Lepetit is with the EPFL/IC/ISIM/CVLab, Station 14, CH-1015 Lausanne, Switzerland. E-mail: vincent.lepetit@epfl.ch.*

Manuscript received 24 Feb. 2010; revised 18 Aug. 2010; accepted 3 Nov. 2010; published online 9 Dec. 2010.
Recommended for acceptance by D. Schmalstieg.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-02-0051.
Digital Object Identifier no. 10.1109/TVCG.2010.262.

2 Related Work

Many multiple object *2D* tracking algorithms have been developed [6], [7]. However, the 2D problem is only remotely related to the *3D* case we address in this paper and we will not discuss it here. By contrast, works on multiple object 3D tracking are rare and most of them are very recent. We will start by quickly reviewing single object 3D tracking methods and then discuss the few existing works on multiple object 3D tracking.

2.1 Single Object Tracking by Detection

A very robust way to track objects is to rely on a wide baseline matching of feature points. The target object can be detected in each input frame independently by matching feature points extracted in the input frame against those extracted in some reference frames (also called keyframes) representing the target objects. This was done, for example, in [2], [8], [9].

Many methods for wide baseline matching, or recognition, of feature points are available [10], [11], but [2] is very simple to implement and very fast. It is based on a classifier

called Ferns that is trained to recognize feature points under image perturbation including perspective change, and our method makes use of this technique for the detection part. It is robust to occlusion, lighting, scale, and perspective changes. However, a naive extension to several objects would not scale, since the system would have to try all the objects for every single input frame, making the complexity grow linearly with the number of objects.

2.2 Frame-to-Frame Tracking

Our method is also related to frame-by-frame 3D tracking approaches, which tracks image features such as edges [12], [13] or feature points [4] over the sequence. In [4], feature points are extracted from the current frame and matched against those from the previous frame, but also against those extracted in keyframes.

We use a similar scheme, however the method of [4] to establish matches with the keyframes is very vulnerable to fast motion, contrary to ours. Another difference is the way we fuse the two sets of matches. Our approach relies on error propagation, which allows proper error handling.

2.3 Multiple Object 3D Tracking

A few works on multiple object 3D tracking appeared recently [14], [15], [16]. [14] introduces a particularly fast matching method based on histograms created during a training phase. It attempts to detect the objects in each frame independently, without exploiting temporal constraints.

By contrast, and similarly to our own work, [15] and [16] combine detection and tracking. In particular, [15] optimizes a score evaluating the trade-off between detection and frame-to-frame tracking, as detection is more robust but requires more time. It also "masks" the parts of the image where an object is already tracked, to avoid running the detection process on these parts and thus save computation time. These different aspects could be integrated in our approach. In this paper, we mostly focus on how to propagate the pose and 3D locations uncertainties in the context of detection and tracking combinations.

3 Multiple Object 3D Tracking

In this section, we first give an overview of our multiple object detection and tracking algorithm, and then detail it step by step.

3.1 Overview

As discussed in the introduction, we want to rely on object detection because it makes tracking applications more robust, but it has to be done carefully to keep processing time consistent with real-time constraints. An overview of our approach is given in Fig. 3. Our starting idea is to distribute the time complexity of detecting multiple objects over consecutive frames. We do not try to detect every object in every input frame but only as many as possible while maintaining camera frame rate. The objects present in the frame but not detected will be detected in one of the next few frames. That results in a small delay that is kept under a second in practice, so it is not really perceptible by the user.

When a new object appears, it is therefore quickly detected by the system which initializes a frame-by-frame tracking for this object. To do so, we rely on what we call "temporal keypoints," which are feature points detected on the object surface and that matched over consecutive frames. From them, we can continue to estimate the object pose accurately even without object detection because it is less time-consuming method compared to object detection. The other advantage is that it is usually more accurate.

In the meantime, we keep trying to detect the application objects, even when a frame-by-frame tracking process is already running for them. That way, we prevent common problems such as loss of track due to fast motion or occlusions, and drift. Note that detection and tracking are clearly separated and most of the computations can be done on different cores of the processor in parallel.

In the next sections, we first detail the data structure we use to represent the objects tracked by the system, then we discuss the detection process and describe how temporal keypoints are used and fused to the object detection results to provide a stable track of the visible objects.

3.2 Object Model

Our object model contains both the geometrical information and appearance of a target object. The geometrical information is a standard CAD 3D model stored as a list of triangles. Many software tools now exist to easily build such 3D models from a set of pictures.

The appearance part is made of a small set of keyframes that are registered images of the object shot from various viewpoints so that they cover most of the object. Usually, 3-4 keyframes are sufficient to cover an object along 360 degrees with some parts taken more than once. In each keyframe, we extract feature points we call keypoints. It is easy to estimate the 3D locations $\mathbf{M}_i$ of these keypoints by back projecting them on the object 3D model. These keypoints and their 3D locations are stored as well. They will be used during the detection phase.

3.3 Sequential Multiple Object Detection

3.3.1 Keyframe Selection

We group the keyframes for all the objects into different subsets. We match each input frame against only one subset, the different subsets being tried in turn. This is a simple method to keep the detection running at frame rate, where the price to pay is a possible short lag between the apparition of an object and its detection by the system.

By contrast with our previous work [5], each subset contains only keyframes from a specific object. This allows us to have a more efficient keyframe detection strategy. A keyframe that is matched successfully with an input frame is likely to also match the next input frames. Such a keyframe is therefore tagged to be tried first the next time an input frame must be matched against its subset. If the match score between this keyframe and the input frame is sufficiently high, we can consider the next subset without checking the other keyframes in the same subset. This approach allows to save computation time without sacrificing robustness most of the time. More details about keyframe matching are described in the next section.

3.3.2 Matching with Keyframes and Pose Estimation

We use the method described in [2] to match the input frame against keyframes because it is fast and robust under

perspective distortion and different viewing conditions. It gives a number of initial correspondences between feature points extracted in the input frame and the keypoints of each of the keyframes in S_i for which the 3D locations $\mathbf{M}_i$ are known. Since some of these correspondences can be erroneous, the object pose is estimated using RANSAC and a noniterative P-n-P algorithm [17].

The detection is considered successful when the number of inliers found by RANSAC is larger than a threshold. If the object was not present before, we perform a nonlinear optimization to refine the object pose computed by RANSAC to initialize the frame-by-frame tracking. If the object was already present, the inlier matches are added to the frame-by-frame matching to estimate the object pose as it will be described below in Section 3.4.

3.3.3 Evenly Distributed Keypoint Extraction

Extracting keypoints for object detection and tracking must be done with care, especially in our case where we want to consider different objects simultaneously. The number of keypoints must be kept reasonably low to maintain efficiency, but enough keypoints must be extracted on an object if the system must detect and track this object. Keeping only the strongest keypoints over the image would not work: as shown in Fig. 4, if two objects are visible, one much more textured than the other one, the less textured object will exhibit much less keypoints, making detection and tracking for this object likely to fail.

Therefore, we adopt a mechanism to evenly distribute the keypoints over the input image. The first step is to extract (much) more than N keypoints, where N is the number of keypoints we actually want to extract. The image is split into $M \times M$ and we keep the $\frac{N}{M^2}$ strongest keypoints in each region. This results into a distribution that is now more suitable to detection and tracking.

3.4 Stable Tracking with Temporal Keypoints

We now explain our stable tracking algorithm that combines object detection and frame-by-frame tracking. Frame-by-frame tracking has two purposes: it completes detection to track the object whenever it is present, and it stabilizes the estimated pose, removing the jittering effect that detection alone would produce.

We extract feature points in each input frame and match them against those extracted in the previous frame using standard techniques based on cross-correlation and local search. Compared to the KLT tracker [18], it allows us to easily handle appearing and disappearing feature points, and to prevent drift of the tracked locations. We call these feature points "temporal keypoints."

3.4.1 Robust Pose Estimation

We fuse temporal keypoints with keypoints matched during object detection to estimate the object poses. For a given object, its pose parameters at time t, $\mathbf{p}^{(t)}$, are estimated by minimizing

$$\mathbf{p}^{(t)} = \arg\min_{\mathbf{p}} \sum_{\mathrm{i}} \rho(\|\phi(\mathbf{p}, \mathbf{M}_{\mathrm{i}}) - \mathbf{m}_{\mathrm{i}}\|^2) + \sum_j w_j \rho(\|\phi(\mathbf{p}, \mathbf{X}_j) - \mathbf{x}_j\|^2), \quad (1)$$

where $\mathbf{p}$ is a 6-vector that contains the rotation and translation values; the $\mathbf{M}_i$ are the 3D locations of the keypoints extracted from the matched keyframe; the $\mathbf{m}_i$, their 2D locations, found by the object detection process explained in Section 3.3.2; and the $\mathbf{X}_j$ and $\mathbf{x}_j$ are, respectively, the 3D and 2D locations found for the jth temporal keypoint. $\phi(\mathbf{p}, \mathbf{M})$ is a function that returns the 2D projection of $\mathbf{M}$ under camera pose $\mathbf{p}$.

The terms of the sums in (1) are weighted differently: the w_js are weights that will allow to handle the fact that the $\mathbf{X}_j$ are not known exactly, as it will be detailed below. Since the 3D locations $\mathbf{M}_i$ come from the keyframes registered offline, we assume they are error-free and we do not use such a weight for the corresponding terms. $\rho(\cdot)$ is a robust estimator; we use an approximation of the Tukey estimator

$$\rho(x) = \begin{cases} x^2 & \text{when } x^2 < c^2 \\ c^2 & \text{otherwise,} \end{cases} \quad (2)$$

where c is a thresholding value. When no correspondences are available from the detection, only the second term of the sum is used. In our implementation, we use the Levenberg-Marquardt algorithm for minimization.

3.4.2 Temporal Keypoint Generation

The 3D locations of the temporal keypoints are estimated by back-projecting their 2D location in the previous image using the estimated pose $\mathbf{p}^{(t-1)}$. Back-projection can be performed efficiently using OpenGL rendering to quickly find the triangle on which the keypoint lies [4]. This way, we can also easily handle occlusion between the tracked objects by assigning the temporal keypoint to the closest object.

Since $\mathbf{p}^{(t-1)}$ is only an approximation of the true pose, we use the w_js to weight the contributions of the different temporal keypoints, according to their error estimates. We take w_j as

$$w_j = \omega(\mathrm{tr}(\mathbf{\Sigma}_{\mathbf{X}_j})), \quad (3)$$

where $\omega(x) = \frac{\rho'(x)}{x}$ is the weight function corresponding to the robust estimator $\rho(.)$, $\mathrm{tr}(\cdot)$ is the trace function, and $\mathbf{\Sigma}_{\mathbf{X}_j}$ is the covariance on the 3D location $\mathbf{X}_j$. $\mathbf{\Sigma}_{\mathbf{X}_j}$ is obtained in two steps: first, we estimate the covariance $\mathbf{\Sigma}_{\mathbf{p}^{(t-1)}}$ of the pose estimate for the previous frame. Then, we propagate this error to $\mathbf{X}_j$.

Under the standard assumption that the correspondences are independent, $\mathbf{\Sigma}_{p(t-1)}$ can be computed by back-propagation as

$$\mathbf{\Sigma}_{\mathbf{p}^{(t-1)}} = \left(\left(\frac{\partial f_k}{\partial \mathbf{p}} \right)^{\top}_{\mathbf{p}^{(t-1)}} \mathbf{\Sigma}_k \left(\frac{\partial f_k}{\partial \mathbf{p}} \right)_{\mathbf{p}^{(t-1)}} \right)^{-1}, \quad (4)$$

where $\mathbf{\Sigma}_k$ is the 2×2 covariance on the extracted keypoints, and the f_k are functions that can be expressed as

$$f_k(\mathbf{p}) = \phi\big(\mathbf{p}, \mathbf{X}_k^{(t-1)}\big). \quad (5)$$

In this expression, $\mathbf{X}_k^{(t-1)}$ denotes the kth temporal keypoint used to estimate pose $\mathbf{p}^{(t-1)}$. Let's now consider a temporal keypoint that comes from a match $\mathbf{x}_j^{(t-1)} \leftrightarrow \mathbf{x}_j$. The covariance $\mathbf{\Sigma}_{\mathbf{X}_j}$ of its 3D location $\mathbf{X}_j$ can be estimated by forward propagation of $\mathbf{\Sigma}_{\mathbf{p}^{(t-1)}}$ as

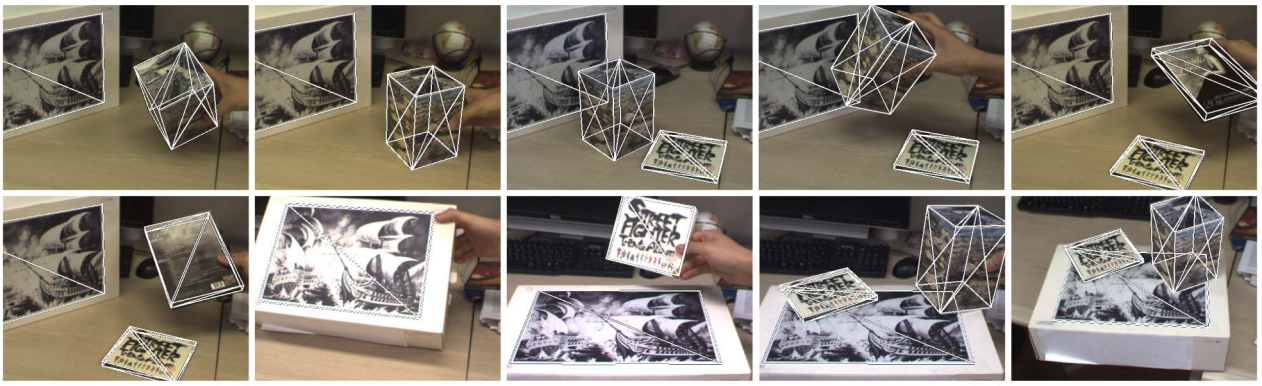

Fig. 1. Multiple 3D object tracking: our method can automatically recognize and track several objects simultaneously under partial occlusions and illumination changes, at about 40 Hz.

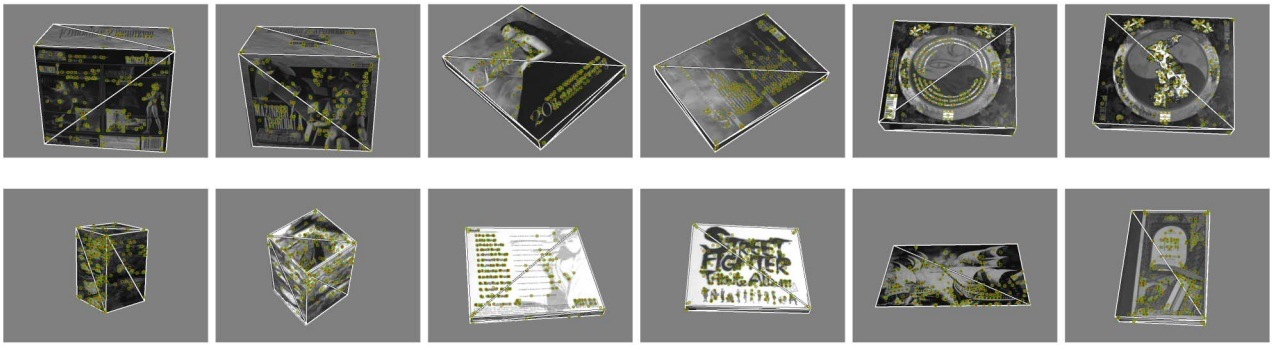

Fig. 2. Keyframes with registered 3D CAD models used in our experiments.

$$\boldsymbol{\Sigma}_{\mathbf{X}_j^{(t)}} = \left(\frac{\partial g_k}{\partial \mathbf{p}}\right)_{\mathbf{p}^{(t-1)},\mathbf{x}_j^{(t-1)}} \boldsymbol{\Sigma}_{\mathbf{p}^{(t-1)}} \left(\frac{\partial g_k}{\partial \mathbf{p}}\right)^{\top}_{\mathbf{p}^{(t-1)},\mathbf{x}_j^{(t-1)}} + \left(\frac{\partial g_k}{\partial \mathbf{x}}\right)_{\mathbf{p}^{(t-1)},\mathbf{x}_j^{(t-1)}} \boldsymbol{\Sigma}_{\mathbf{p}^{(t-1)}} \left(\frac{\partial g_k}{\partial \mathbf{x}}\right)^{\top}_{\mathbf{p}^{(t-1)},\mathbf{x}_j^{(t-1)}} \tag{6}$$

where $g_k(\mathbf{p}, \mathbf{x})$ is a function that back-projects keypoint $\mathbf{x}_j^{(t-1)}$ on the object under pose $\mathbf{p}$. Its Jacobian matrices in (6) are computed at pose $\mathbf{p}^{(t-1)}$ and for keypoint $\mathbf{x}_j^{(t-1)}$.

3.4.3 Temporal Keypoint Verification

Temporal keypoints that lie on an unknown object occluding a tracked object may cause drift even when using robust estimation. We therefore use an extra step after the frame-to-frame tracking stage.

This verification is done by predicting the appearance of the point using the object pose estimated for the previous image and its appearance in the keyframe. Then, the predicted appearance is computed with the input image using Normalized Cross Correlation. If the NCC score is higher than a predefined threshold, the point is kept; otherwise, it is likely to lie on an occluding object and it is discarded.

We first select the keyframe having the closest orientation to the estimated object pose in the input image and use this keyframe to predict the appearance of each temporal keypoint in the area of interest as shown in Fig. 5. We generate four 3D points $\mathbf{c}_i$ that define a 3D square centered on the keypoint 3D location $\mathbf{X}_j$ and tangent to the object surface. To do that, we consider a first vector $\mathbf{P}_0$ which origin corresponds to the keypoint and which points toward one of the vertices of the mesh triangle the keypoint lies on: $\mathbf{P}_0 = \mathbf{V}_0 - \mathbf{X}_j$. An orthogonal vector that lies on the same triangle can be obtained by computing the cross-product of $\mathbf{P}_0$ and the vector normal $\mathbf{n}$ to the triangle: $\mathbf{P}_1 = \mathbf{n} \times \mathbf{P}_0$. The $\mathbf{c}_i$ points can finally be created as

$$\mathbf{c}_i = \mathbf{X}_j + s(\pm\mathbf{P}_0 \pm \mathbf{P}_1), \tag{7}$$

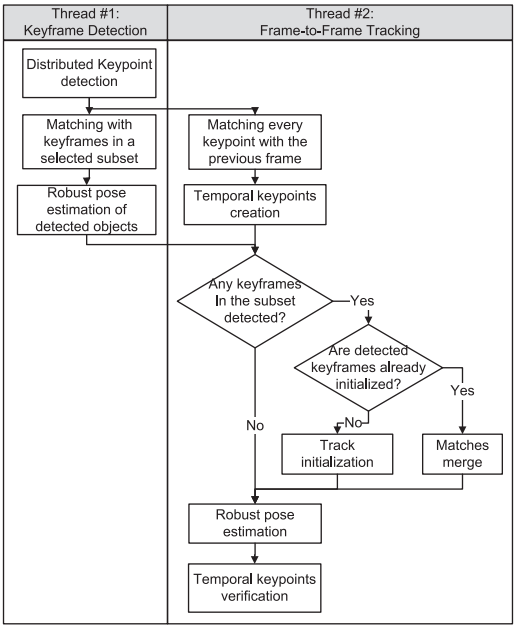

Fig. 3. Overview of the processing for one input frame. Detection and frame-by-frame tracking are performed on two different cores in parallel.

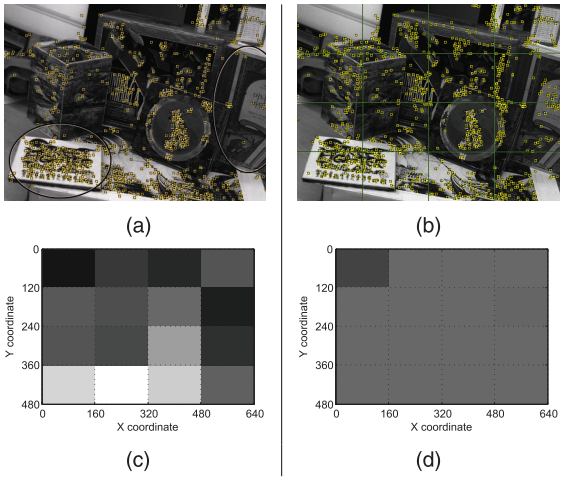

Fig. 4. Extracting evenly distributed keypoints. $N = 1{,}000$ keypoints extracted: (a) When keeping the strongest keypoints over the whole image. Most of the keypoints are clustered at the bottom of the image in this example. (b) When first splitting the image into $M \times M$ regions, and keeping the $\frac{N}{M^2}$ strongest keypoints in each region. The keypoints distribution is now more suitable to detection and tracking. (c) and (d) shows the density of the keypoints in the regions by the two methods.

where s is a scaling factor which determines the size of the patch.

Once we have the 3D coordinates $\{\mathbf{c}_i\}$ of the patch, we project them into both the keyframe and the input image and compare the corresponding 2D patches with NCC. The temporal keypoint is kept as valid only if the NCC between the patches is higher than a predetermined threshold.

3.5 Thread Synchronization

The detection and tracking threads require a correct synchronization if we do not want to lose the advantage of the two processes running in parallel. In our previous work [5], we used a simple synchronization where both threads begin on a frame feed as shown in Fig. 6a. It is actually possible to improve the synchronization so that each thread advance to the following process without suspending as shown in Fig. 6b. This scheme is of a different nature than [19] because in our case both threads are designed to respond to every frame.

After keyframe matching, the detection thread sends the detection result to the tracking thread and moves on to the next frame. While the detection thread is grabbing and matching the next image ($\mathbf{I}_{t+1}$), the tracking thread is working with the current image ($\mathbf{I}_t$). This asynchronous process causes no negative effect since the tracking thread completes before the keyframe detection of the next image,

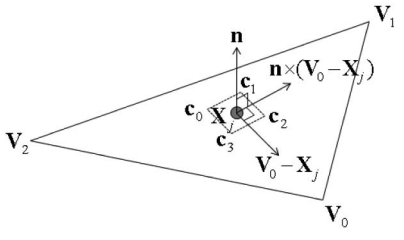

Fig. 5. Creation of a patch for temporal keypoint verification. $\mathbf{c_0}$-$\mathbf{c_3}$ are the patch corners on the model surface.

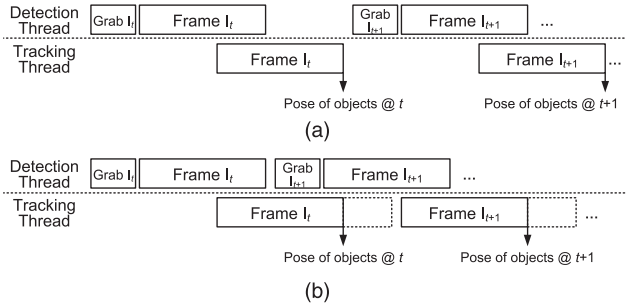

Fig. 6. Thread synchronization schemes used (a) in our previous work [5] and (b) in this paper. (a) Both detection and tracking starts at a new frame feed. (b) The threads advance to the next frame processing without suspension.

and gives more time for frame-to-frame tracking as shown in Fig. 6b as dotted areas.

4 Experiments

In order to evaluate our approach, we considered the seven objects shown Fig. 2. For each object, we used one or two keyframes to cover the both sides or large view difference of the objects. The models and the objects in the keyframes were registered by manual 3D-2D correspondences. More complex shaped objects could also be considered if their 3D model is available.

We tested our implementation on a 3.2 GHz multicore CPU PC. It works at around 40 frames per second, which is almost the same as when we run the detection only [2] despite of the combined tracking steps. It also handles multiple objects and improves stability, and more importantly, the tracking frame rate does not decrease significantly with the number of considered objects.

In Fig. 1, up to three objects are in the field of view of the camera and tracked simultaneously. The pose of the objects is accurately estimated despite the partial occlusions between objects. More difficult conditions were tested in the sequence of Fig. 7. Up to five objects appear in the image and occlude each other. Sometimes, only a very small part of the object is visible but the object is still tracked successfully. Fig. 8 shows another sequence where one object is completely occluded by another one but tracked again when it is visible again.

4.1 Accuracy

Fig. 9 demonstrates the accuracy of the proposed tracking algorithm. We fixed a camera and tracked the movement of an object which was constrained to lie on a planar surface as shown on the first row. The recovered trajectory for the object centroid is shown on the second row and exhibits only very limited jitter and lies on a 3D plane as expected. This validates the accuracy of the system.

4.2 Comparison with Detection Only

Fig. 10 compares the results obtained with the proposed method and the results obtained with detection only on the sequence of Fig. 7. For clarity, the graph shows only one coordinate of the estimated trajectory of the objects. Our method improves the stability of the trajectory and improves robustness because the tracking is maintained even when the detection fails.

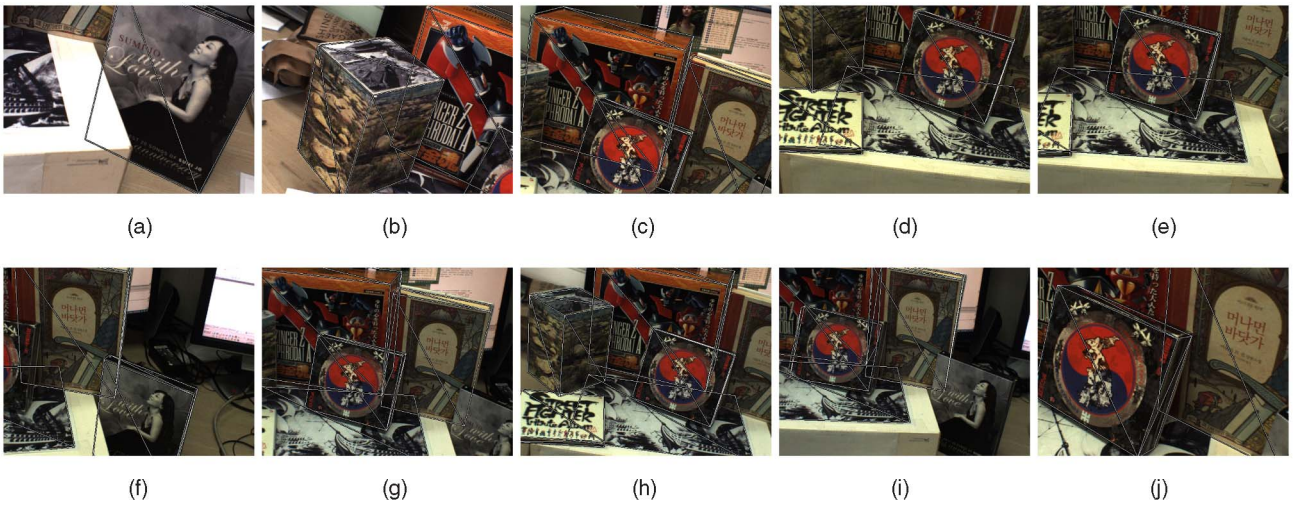

Fig. 7. The sequence used for detection and tracking performance evaluation. The 3D CAD models are projected with estimated poses by our tracking. Up to five objects appear in the same frame. (b) and (e) show that the proposed method can correctly estimate the object pose even when small part of them is visible.

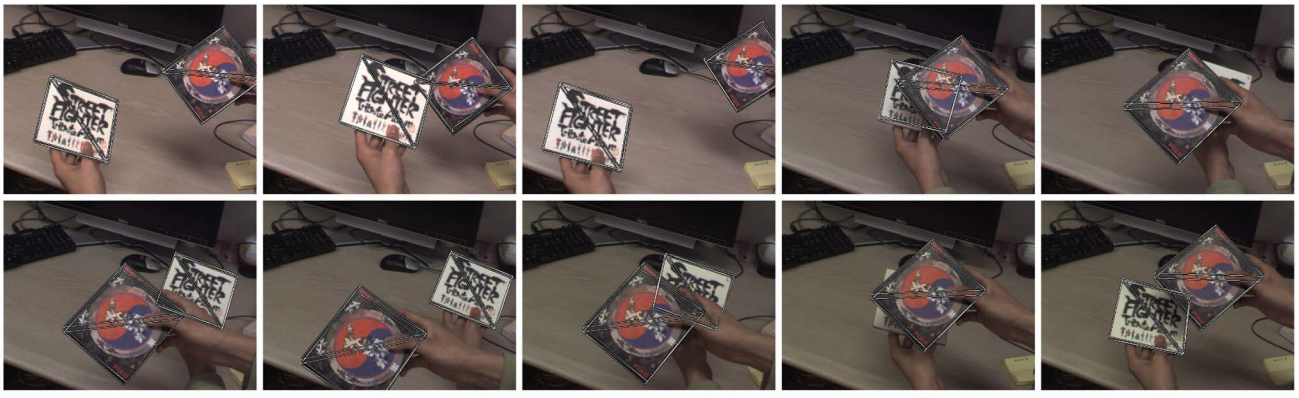

Fig. 8. Tracking several objects under difficult conditions: the objects are moving fast, one object completely occludes the other one in the middle of the sequence, but the occluded object is re-initialized when it is visible again.

4.3 Occlusion

The contribution of the extra step of temporal keypoint verification described in Section 3.4.3 is presented in Fig. 11.

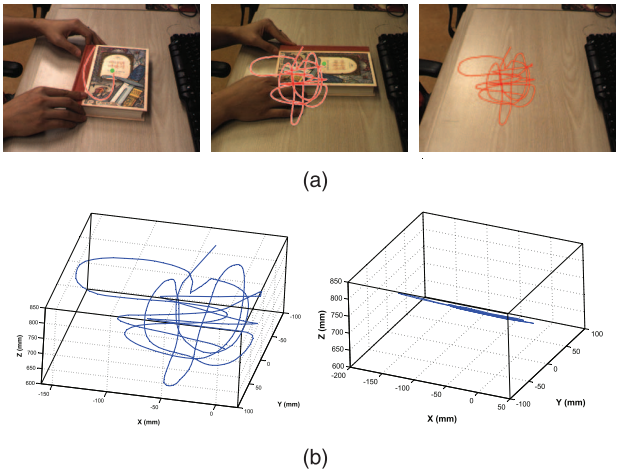

Fig. 9. Accuracy of the proposed tracking algorithm. (a) An object is moved while constrained to lie on a planar surface. (b) The recovered trajectory for the object centroid exhibits only small jitter and lies on a 3D plane as expected.

The temporal keypoints which have undergone the verification step are classified into valid and invalid keypoints as shown in the second column of Fig. 11. When we consider only valid temporal keypoints, the tracking becomes robust and works correctly as the last column in Fig. 11.

4.4 Frame Rate

Fig. 12 plots the evolution of frame rate for the sequence shown in Fig. 7. While the number of tracked objects varies over the sequence, the frame rate of keyframe detection is almost constant around 40 Hz which is close to a single object detection. This is achieved by the subset approach in Section 3.3.2. The frame-to-frame tracking is always faster than the detection even with every detected object. This shows that our approach is suitable for interactive applications and does not degrade with the number of tracked objects as long as this number remains reasonable.

4.5 Detailed Computation Times

Table 1 shows the performance measurement of the keyframe detection and frame-to-frame tracking threads. The time was measured through the image sequence shown in Fig. 7. The frame-to-frame tracking had to consider up to

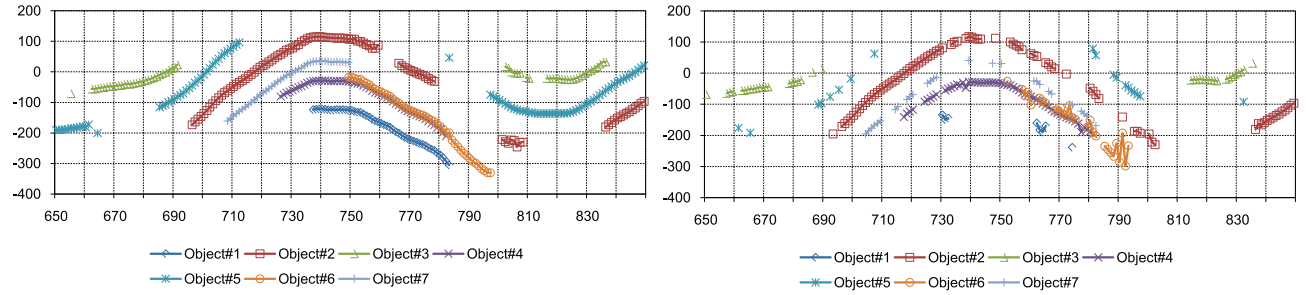

Fig. 10. Estimated poses of the objects in the sequence shown in Fig. 7 by the proposed method (a) and by detection only (b). For visibility, the graphs show only one coordinate of the position of each object. For the detection only case, from frame #690-frame #710, object #5 is rarely detected and the pose of object #6 is very unstable between the 785th and 795th frames. By contrast, our method retrieves longer and smoother trajectories.

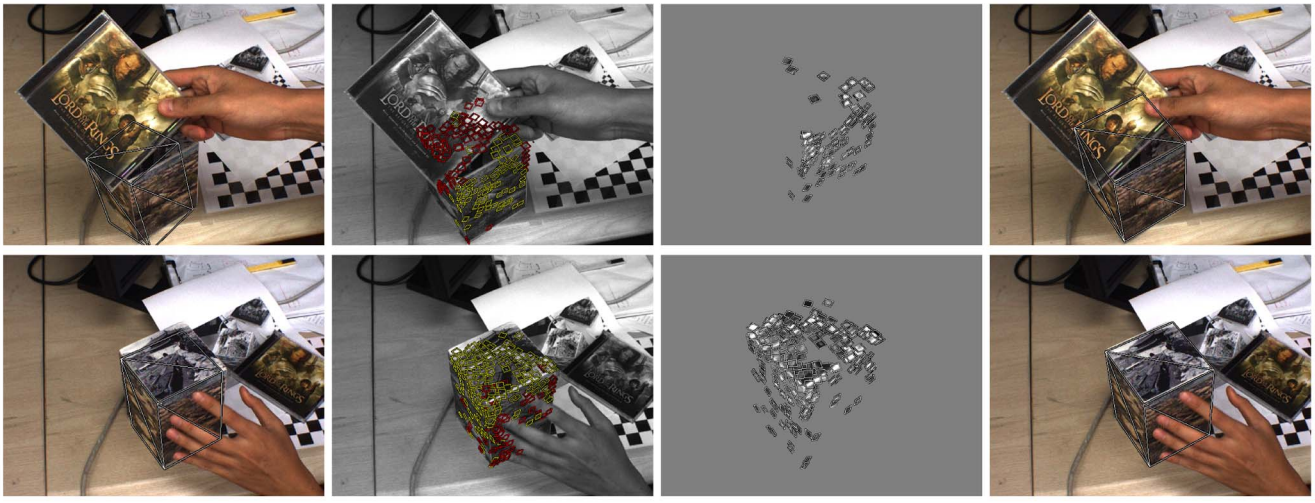

Fig. 11. Effect of occlusion by unknown objects and temporal keypoint verification. First column: without the verification step, frame-to-frame tracking is prone to drift in case of occlusion. Second column: the temporal keypoints verified as correct are marked with yellow squares and those found incorrect are marked with red ones. Third column: the corresponding keypoints to the second column show only the correct points in the keyframe. Last column: tracking without invalid temporal keypoints.

five objects in one input image. In Thread #1, the keypoint detection includes color-to-gray conversion of the input image, image pyramid construction, and keypoint extraction. The other step covers keypoint matching with the current subgroup of keyframes and pose estimation with outlier rejection. Frame-to-frame tracking step in Thread #2 takes all of the other steps except for temporal keypoint verification in the frame-to-frame tracking.

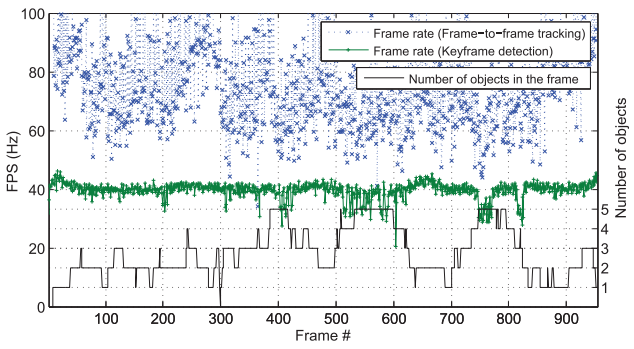

Fig. 12. Tracking frame rate: the green and blue curves represent the evolution of the frame rate over a period of time, the black curve represents the number of objects tracked by the system. While the number of objects varies, the frame rate for the detection remains almost constant and the frame-to-frame tracking performs always faster than the detection.

5 Conclusion

We presented a method for tracking simultaneously multiple 3D objects using a monocular camera. Compared to our preliminary work [5], this paper presented several technical improvements that contributed to the improved frame rate and robustness to occlusion. Typical applications are tabletop AR and tangible interaction applications.

However, our approach still not properly scales with the number of objects and is currently limited to a database with a few tens of objects. To handle larger databases, it

TABLE 1
Average Processing Time of Detection and Tracking

	Step	Average time (ms)
Thread #1 (Detection)	Keypoint detection	12.66
	Matching & pose estimation	12.61
Thread #2 (Frame-to-Frame Tracking)	Frame-to-frame tracking	2.77
	Track (re-)initialization	0.27
	Temporal keypoint verification	4.99

may be interesting to first limit the number of possible present objects with a scalable recognition method such as [20] which does not provide any exact pose but can provide the list of the visible objects.

ACKNOWLEDGMENTS

This research is supported by the Ministry of Culture, Sports and Tourism (MCST) and the Korea Creative Content Agency (KOCCA), under the Culture Technology (CT) Research and Development Program 2010.

REFERENCES

[1] V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects," *Foundations and Trends in Computer Graphics and Vision,* vol. 1, no. 1, pp. 1-89, 2005.
[2] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast Keypoint Recognition Using Random Ferns," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 32, no. 3, pp. 448-461, Mar. 2010.
[3] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *Proc. Int'l Symp. Mixed and Augmented Reality,* Nov. 2007.
[4] L. Vacchetti, V. Lepetit, and P. Fua, "Stable Real-Time 3D Tracking Using Online and Offline Information," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 10, pp. 1391-1402, Oct. 2004.
[5] Y. Park, V. Lepetit, and W. Woo, "Multiple 3D Object Tracking for Augmented Reality," *Proc. Int'l Symp. Mixed and Augmented Reality,* Sept. 2008.
[6] D. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Trans. Automatic Control,* vol. 24, no. 6, pp. 843-854, Dec. 1979.
[7] S. Oh, S. Russell, and S. Sastry, "Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems," *Proc. IEEE Conf. Decision and Control,* Dec. 2004.
[8] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile Phones," *Proc. Int'l Symp. Mixed and Augmented Reality,* Sept. 2008.
[9] I. Skrypnyk and D. Lowe, "Scene Modelling, Recognition and Tracking with Invariant Image Features," *Proc. Int'l Symp. Mixed and Augmented Reality,* Nov. 2004.
[10] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.
[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding,* vol. 110, no. 3, pp. 346-359, 2008.
[12] T. Drummond and R. Cipolla, "Real-Time Visual Tracking of Complex Structures," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 932-946, July 2002.
[13] A. Comport, E. Marchand, M. Pressigout, and F. Chaumette, "Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 4, pp. 615-628, July 2006.
[14] S. Taylor and T. Drummond, "Multiple Target Localisation at over 100 FPS," *Proc. British Machine Vision Conf.,* Sept. 2009.
[15] D. Wagner, D. Schmalstieg, and H. Bischof, "Multiple Target Detection and Tracking with Guaranteed Framerates on Mobile Phones," *Proc. Int'l Symp. Mixed and Augmented Reality,* Oct. 2009.
[16] J. Pilet and H. Saito, "Virtually Augmenting Hundreds of Real Pictures: An Approach Based on Learning, Retrieval, and Tracking," *Proc. IEEE Virtual Reality,* Mar. 2010.
[17] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EP*n*P: An Accurate *O(n)* Solution to the P*n*P Problem," *Int'l J. Computer Vision,* vol. 81, no. 2, pp. 155-166, Feb. 2009.
[18] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," technical report, Carnegie Mellon Univ., Apr. 1991.
[19] T. Lee and T. Hollerer, "Multithreaded Hybrid Feature Tracking for Markerless Augmented Reality," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 3, pp. 355-368, May 2009.
[20] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," *Proc. Conf. Computer Vision and Pattern Recognition,* pp. 2161-2168, Oct. 2006.

Youngmin Park received the BS degree in computer engineering from the Kangwon National University, Gangwon-do, Korea, in 2004, and the MS degree from the Department of Information and Communications (DIC), Gwangju Institute of Science and Technology (GIST), Gwangju, Korea, in 2006, where he is currently working toward the PhD degree. His research interests include augmented reality, 3D computer vision, image processing, and mobile computing. He is a student member of the IEEE.

Vincent Lepetit received the engineering and master degrees in computer science from the ESIAL in 1996, and the PhD degree in computer vision in 2001, all from the University of Nancy, France, after working in the ISA INRIA team. He is a senior researcher at the Computer Vision Laboratory, EPFL, Laussane, Switzerland. He then joined the Virtual Reality Lab at the EPFL (Swiss Federal Institute of Technology) as a postdoctoral fellow and became a founding member of the Computer Vision Laboratory. His research interests include vision-based Augmented Reality, 3D camera tracking, object recognition, and 3D reconstruction.

Woontack Woo received the BS degree in electronics engineering from the Kyungpook National University, Daegu, Korea, in 1989, the MS degree in electronics and electrical engineering from the POSTECH, Pohang, Korea, in 1991, and the PhD degree in electrical engineering systems from the University of Southern California (USC), in 1998. In 1999, as an invited researcher, he joined Advanced Telecommunications Research (ATR), Kyoto, Japan. Since February 2001, he has been with the Gwangju Institute of Science and Technology (GIST), Gwangju, Korea, where he is an associate professor in the Department of Information and Communications (DIC) and director of the Culture Technology Institute (CTI). His research interests include 3D computer vision and its applications including attentive AR and mediated reality, HCI, effective sensing, and context-aware for ubiquitous computing. He is a member of the IEEE and the IEEE Computer Society.