Noname manuscript No.
(will be inserted by the editor)

Scalable Real-time Planar Targets Tracking for Digilog Books

Kiyoung Kim · Vincent Lepetit · Woontack Woo

Received: date / Accepted: date

Abstract We propose a novel 3D tracking method that supports several hundreds of pre-trained potential planar targets without losing real-time performance. This goes well beyond the state-of-the-art, and to reach this level of performances, two threads run in parallel: The foreground thread tracks feature points from frame-to-frame to ensure real-time performances, while a background thread aims at recognizing the visible targets and estimating their poses. The latter relies on a coarse-to-fine approach: Assuming that one target is visible at a time, which is reasonable for digilog books applications, it first recognizes the visible target with an image retrieval algorithm, then matches feature points between the target and the input image to estimate the target pose. This background thread is more demanding than the foreground one, and is therefore several times slower. We therefore propose a simple but effective mechanism for the background thread to communicate its results to the foreground thread without lag. Our implementation runs at more than 125 frames per second, with 314 potential planar targets. Its applicability is demonstrated with an Augmented Reality book application.

This research is supported by MCST and KOCCA, under the CT R&D Program 2010.

Kiyoung Kim
GIST U-VR Lab., S. Korea
Tel.: +82-62-9703157
Fax: +82-62-9702204
E-mail: kkim@gist.ac.kr

Vincent Lepetit
EPFL CVLab, Switzerland
E-mail: vincent.lepetit@epfl.ch

Woontack Woo
GIST U-VR Lab., S. Korea
E-mail: wwoo@gist.ac.kr

Keywords Planar Target Tracking · Augmented Reality · Digilog Book · Vocabulary Tree

1 Introduction

Many recent Augmented Reality (AR) works have shown that considering only planar objects as tracking targets is enough for many applications. In particular, AR books [2, 17, 15] or digilog books [5, 13] are probably the most representative and latest application of emerging AR edutainment markets and assume the book pages to be rigid and planar.

However, even with this simplification, there is still no satisfying tracking method to handle books with a large number of pages. Early magic books [2] used ARToolkit markers, unfortunately, the markers distract the users' immersion and are fragile to occlusions. More recently, several magic books based on natural features were developed [17, 15], but they usually have high computational costs. Moreover, fast methods such as [12] require a lot of training time and memory and scales very badly with the number of targets.

In this paper, we propose a new scalable marker-less tracker. Target detection and feature tracking run in parallel respectively in a background and a foreground threads to achieve real-time performance and scalability. The foreground thread tracks feature points from frame-to-frame to ensure real-time performances, while a background thread aims at recognizing the visible targets and estimating their poses. The latter relies on a coarse-to-fine approach: Assuming that one target is visible at a time, which is reasonable for digilog books applications, it first uses a vocabulary tree-based image retrieval algorithm [11] to recognize the visible target. It then matches SIFT keypoints between the target and the input image using a kd-tree [9] to estimate the target pose. In addition, the extraction of SIFT keypoint is done on the Graphic Processing Unit (GPU) for speeding up.

The background thread is more demanding than the foreground one, and is therefore several times slower. This could be problematic if this is not taken care of properly. For example, if the foreground thread had to wait for the background thread, that would result in drastic loss of performances. We therefore propose a simple but effective mechanism for the background thread to communicate its results to the foreground thread without lag.

That allows our implementation to run at more than 125 frames per second, with 314 potential planar targets, and we demonstrate its applicability on Augmented Reality magic book application.

In the remainder of the paper, we first address the related work in detail in Section 2. We then describe our method in Section 3. The experimental results are discussed in Section 4, and our digilog book application as an example system is presented in Section 5. Finally, we conclude the paper and discuss about future work in Section 6.

2 Related Work

Many vision-based trackers for AR applications have been suggested in the literature, and can be divided in two categories: marker-based and markerless methods. We review them below in terms of scalability and real-time performance for developing practical AR applications.

2.1 Marker-Based Methods

ARToolKit [6] is one of the early stage trackers widely used in AR applications. It relies on intensity thresholding to detect the markers, and template matching to recognize them. The results were relatively reliable in practice, but the marker recognition procedure was linear in the number of markers, and therefore the performances could drop when the application at hand used many markers. ARToolKitPlus [19] and ARTag [3] for example overcome this scalability issue by using a bar code-like system to encode the marker index in its appearance. But the main drawback remains: The presence of markers on each target object reduces the users' immersion.

2.2 Markerless Tracking Methods

Many markerless, or natural features-based methods have been proposed, but it is now clear that feature point recognition is the key to make applications robust and autonomous in practice. Local descriptors such as SIFT [9] or SURF [1] have been used but do not did not show enough performance for real-time AR applications due to their heavy computational costs. Randomized Trees (RT) [8] and Ferns [12] were suggested to overcome the low speed of the SIFT-like methods. However, the codes provided by [8, 12] required a training time of more than one minute per target. In addition, they do not scale with the number of objects, as the memory consumption is linear with the number of objects.

If the ability to recognize feature point is required, it may cause jittering when used alone because the features are not detected continuously over a camera stream. As a result, a hybrid method, which combines detection and frame-to-frame tracking, is probably required [14, 7, 20]. [20] proposed an efficient method that controls the detection and the tracking tasks dynamically to guarantee a real-time frame-rate. [7] used multicore programming with detection and tracking run on two different cores.

However, to the best of our knowledge, all the existing methods were designed for supporting a single target or less than 10 targets, while many applications, such as AR museum guidance or AR magic books, require the capability to consider more targets. Our main contribution is to show how to consider a large number of potential targets without losing frame-rate.

3 Proposed Method

The overall procedure of the proposed tracker is shown in Figure 1. As already mentioned in the introduction, the tracker consists of two modules, a "recognition module" and a "tracking module". The detection module is in charge of recognizing the visible targets and estimating its pose. It then sends this information to the tracking module, which tracks the recognized target from frame to frame. For efficiency, the two modules run in two separated threads.

3.1 Detection Module

The detection module is based on recent very efficient techniques for image retrieval to recognize the target, and another very efficient data structure for feature point matching.

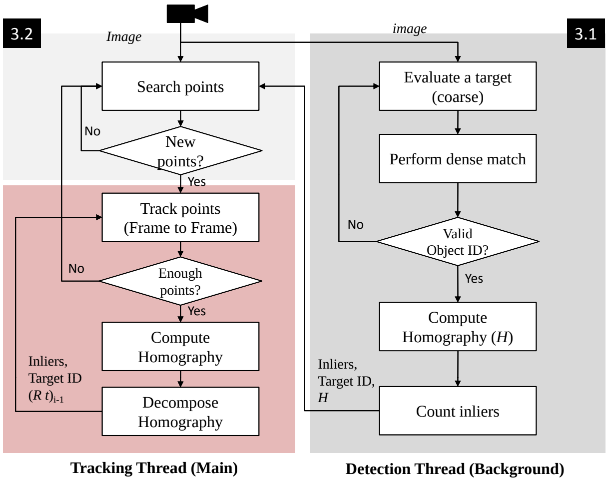

Fig. 1: Overall flowchart of the proposed tracker using two threads.

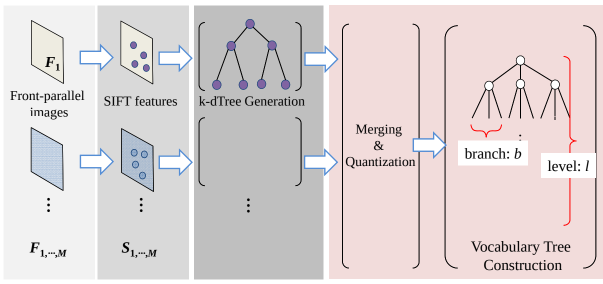

Fig. 2: Procedure illustration of a vocabulary tree construction in off-line preprocessing: M front-parallel images (**F**), M kd-tree with SIFT keypoints (**S**), and one Vocabulary tree.

3.1.1 Preprocessing

We first extract SIFT features in all the reference images of the targets. We then construct a vocabulary tree to quantize these features, as described in [11]. This vocabulary tree will allows us to very quickly recognize the target present in a given input image. The second data structure we build is a kd-tree—one tree per target—to store the SIFT features extracted in the reference image for the target.

3.1.2 Online Detection

For a given input image, we first extract SIFT features from it and run the algorithm described in [11] for image retrieval. The result is a list of reference images of the targets, sorted by similarity with the input image. This is very fast, even with a very large number of targets. The SIFT descriptor is rectified for rotation and scale, and invariant to perspective to some extent, so the method can recognize the targets under large transformations.

Instead of considering only the best response, we keep the first two best candidates. We found in practice that helps to solve the ambiguous case where two targets have many similar features.

To find the correct target, and its pose, we match the SIFT features extracted in the input image against those in the reference images of the two candidates using their associated kd-trees. We then use RANSAC to robustly compute the homography between the input image and each of the two reference images from these matches. The target that gives the largest number of inliers is kept as the correct one. Finally, the camera rotation and translation can easily by computed from the homography and the internal parameters of the camera.

The pseudo-code is given in Algorithm 1. Typically the processing time in the critical section should be shorter than any other modules to maximize the advantage of the multi-core programming. Thus, in the **Write** function, we adjust boolean flags of the shared variables or copy indexes instead of copying variables. At the end, the time for object detection is the sum of the time for the feature extraction, vocabulary tree searching, kd-tree searching with two candidates, and outlier rejection processes.

Algorithm 1 Pseudo-code for real-time detection

while (Tracker is running) and $\mathcal{I}_\mathcal{S} \neq$ **NULL do**
  S_c = **ExtractSIFTFeatures**($\mathcal{I}_\mathcal{S}$);
    // Extract SIFT features from the input image
  (ϕ_1, ϕ_2) = **FindCandidates**(S_c);
    // Find two candidates by image retrieval
  $\phi^* =$ **CountMatch**(ϕ_1) $>$ **CountMatch**(ϕ_2) ? $\phi_1 : \phi_2$;
    // Select the best one
  (H, x_r, H_{err}) = **RejectOutlier**(ϕ^*);
    // Reject outliers by computing the homography H, Inliers, and Error
  if $H_{err} < 4.0$ **then**
    Enter_Critical_Section // Access to the shared variables
      Write $(H, x_r, \mathcal{I}_\mathcal{S})$;
    Leave_Critical_Section
  end if
end while

3.2 Tracking Module

Figure 3 illustrates the relations between the different components of the tracking module and the detection module. We detail them below.

3.2.1 Information from the Detection Module

When the detection module finished to recognize the visible target and to compute its pose, this information

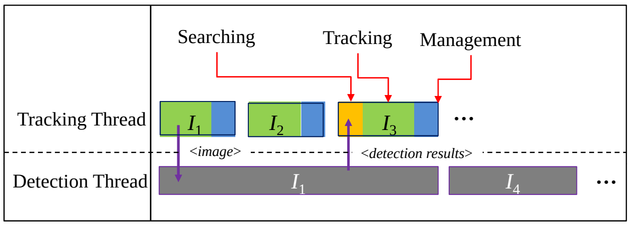

Fig. 3: Illustration of the tracking and the detection on each thread

can be used by the tracking module. Unfortunately, as shown in Figure 3, the detection module is typically slower than the tracking thread. As a result, the tracking module already processes an image captured after the image processed by the detection module.

To compensate, we match the features extracted in the two images. Because the motion between the two images is typically not large, we can use a fast and simple procedure based on cross-correlation and bounded search regions, as described in [16] for example. In practice we use 16×16 correlation windows.

3.2.2 Frame-to-Frame Matching

Once the target is recognize, we can continuously track its feature points over consecutive frames. We use for that a procedure similar to the one described in the previous subsection. The only difference is that we can afford here to use smaller correlation windows and thus speeding up the computations as the images are typically closer.

3.2.3 Features Management

The searching process is not done for every incoming frame. It is called only when new feature points are available by examining whether they are in the tracking queue or not. To avoid slowing down the applications, at most 50 feature points are added to the tracking queue at the same time.

A possible pitfall is to introduce points that are difficult to match and could make the tracking fail. Therefore, we rely on the Sum of the Squared Differences (SSD) between patches in the current and the previous frames to control the quality at every frame. The patch quality is divided into three groups: Good, Neutral, and Bad based on SSD scores. If the feature is bad, it will be replaced by a new one at the next time step. If the feature is neutral, it is replaced only if the newly added feature has a better quality. We fix the maximum number of the tracked features to 300 for real-time performance.

Table 1: Time measurement for preprocessing.

Stage	Time (ms)
Extracting SIFT features	33.254
Building kd-tree	25.712
Building a vocabulary tree	32.842

4 Experimental Results

We evaluated the proposed method with various planar objects. We used a FleaMV camera, which provides 640×480 images at 60FPS. All experiments were performed with Quad-core CPU 2.9 GHz and a nVidia Geforce GTX 285 graphic card. We used OpenCV to implement the proposed algorithm. We also used Sift-GPU [1] and the implementation of vocabulary tree described in [4].

Figure 4 shows the robustness of the system to various challenging imaging conditions: Scale changes, tilt, occlusions, and clutter. Thanks to the detection module, the system can initialize and recover automatically after fast motions, and thanks to the tracking module, it can handle large tilt and occlusions.

As shown in Figure 6, we used 314 pages of an illustrated calendar as target objects. The system scales remarkably well, and there is no loss of frame rate performances when increasing the number of potential targets.

We measured the time for preprocessing. Table 1 shows the measured time for each stage. The average time for SIFT features extraction from one image took $33.254ms$. The average time of building the kd-trees for the 314 targets was $25.712ms$. At the end, the total number of feature points were 291,800. Each target has the number of features from at least 564 and at most 2283. Finally, the time to build the vocabulary tree was $32.842ms$. Thus, $M \times 25.712 + 32.842ms$ were required for M targets in our experimental setup.

We measured the time for the online process as well, which includes the detection and the tracking. For the detection module, we compared four approaches. The first method is a naive way to find out the target by trying each of the 314 possibilities, sequentially. kd-trees of objects were used for matching. The second method is only using kd-tree for matching instead of using a vocabulary tree. All features are used to build one kd-tree in this case. The third method is to use the randomized kd-tree (FLANN) proposed in [10]. The fourth method is our proposed method. The same image sequence was used for the experiments. Not surprisingly, the first method did not work when the number of objects was more than 10. In practice, the detection

[1] http://www.cs.unc.edu/-ccwu/siftgpu/

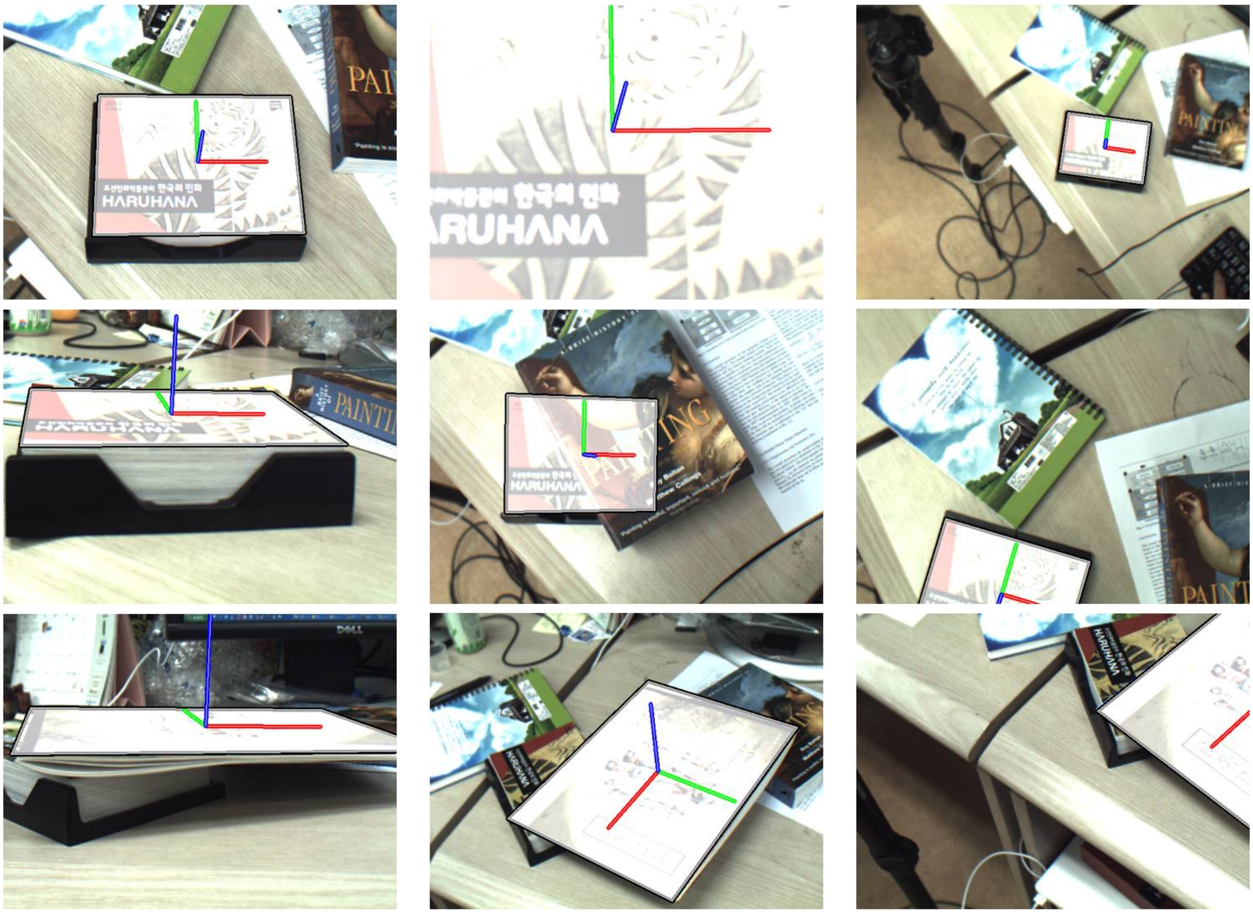

Fig. 4: Pose estimation results from different challenging conditions: Scale changes, tilt, occlusions, and clutter. For visualisation purposes, the target is augmented with a virtual local coordinates system.

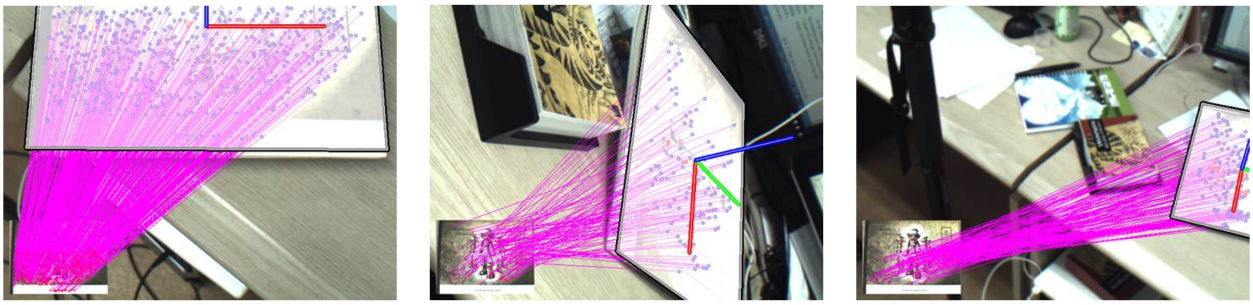

Fig. 5: Visualisation of the feature matches retrieved by our tracker.

could not follow the change of the scene. As a result, the proposed method obtained the most inliers than other methods as shown in Figure 7a. Our approach had twice more inliers than the kd-tree only method. The large number of the inliers helps the tracking stable. The detection time did not show large differences among the three methods as shown in Figure 7b. However, the kd-tree and FLANN methods required a large amount of memory to store their data structure, and we had to reduce the number of the target objects from 314 to 200 in this experiment to make the comparison possible.

In the tracking module, we measured time for the searching, the tracking, and the computing pose. Table 2 shows the average time for each module. As a result, the application can run at more than 125FPS.

Figure 8 shows the measurement results of the real sequence. We observe that the searching process is not carried out at every frame. The reprojection error was less than 2.0 pixels with no visible jitter.

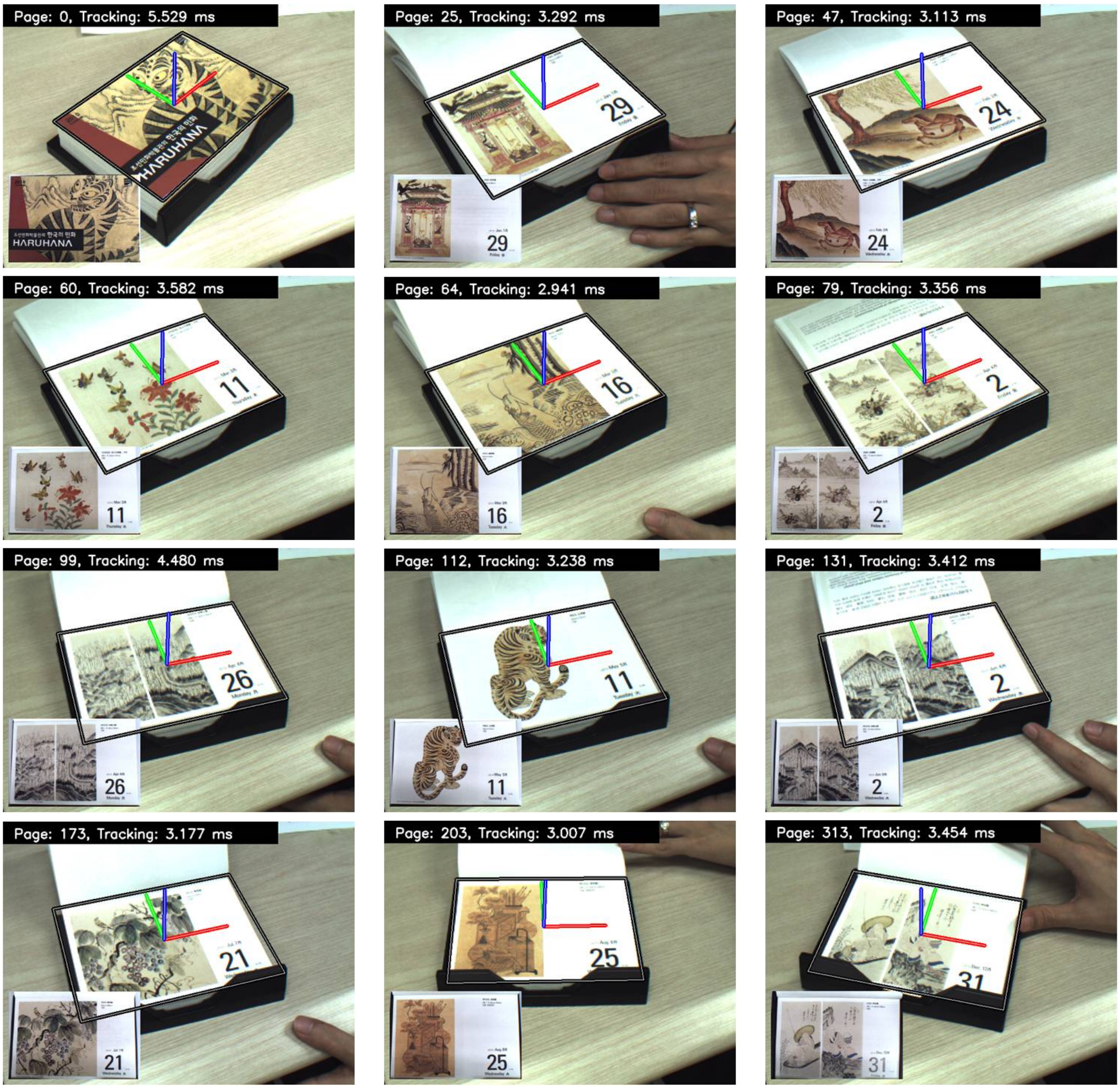

Fig. 6: Snapshots of an application with 314 potential targets: The system scales remarkably well, and there is no loss of frame rate performances when increasing the number of potential targets.

Table 2: Time measurements for the tracking module.

On-line Process	Time (ms)
Searching time (AVG.)	1.257
Frame-to-Frame matching	3.286
Computing / Decomposing Homography	0.982

Finally, we measured how the application is influenced according to the size of the feature set. We measured the detection and the tracking times for different numbers of targets, by randomly choosing the targets.

We successively considered 2, 100, 200, and 314 targets, with respectively 2'258, 100'409, 186'275 and 291'800 feature points for detection. The depth of the vocabulary tree was adjusted from 3 to 6 according to the number of feature points. The experiments were performed over the same sequence.

Figure 9 shows the results. Once the detection was done, the detection speed did not make the feature tracking slower because the tracker can still rely on feature points from the previous frame. The relocalization was affected by the size of the feature set but only marginally: The experiment with 314 targets missed only 4 frames more than the one with 2 targets. The difference on the detection time as shown in Figure 9c did not make a significant differences for the applica-

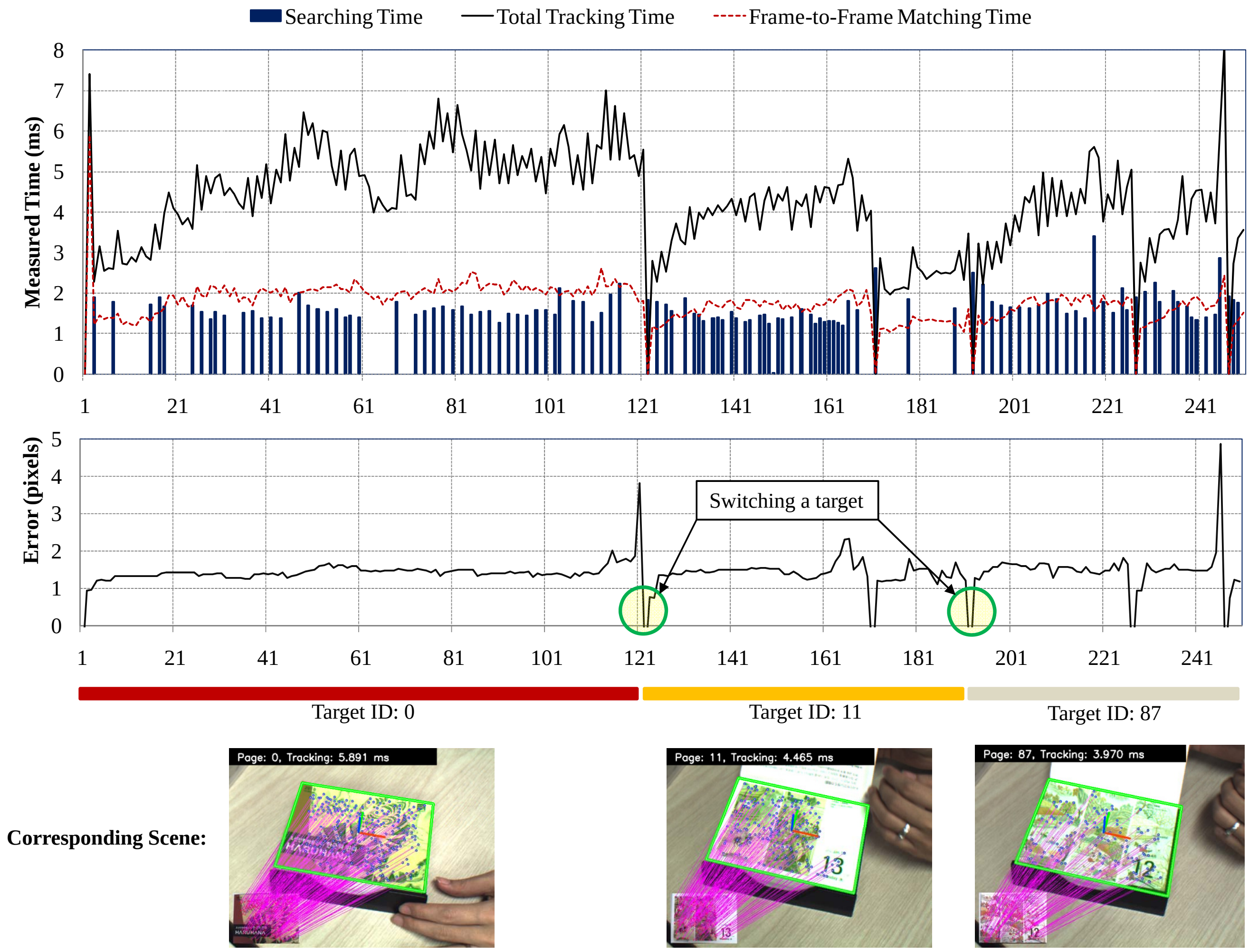

Fig. 8: Performance results of the proposed tracker on the image sequence. (top) Overall tracking time, frame-to-frame matching and searching time described in Section 3.2. Note that the searching module was not called at each frame. (middle) Reprojection errors of the corresponding sequence. (bottom) Snapshots of the corresponding image sequence.

tion: We could track the target within 4 to 8*ms* in all cases.

5 Digilog Book Application

In this section, we describe a digilog book application based on our method.

A Digilog book is an Augmented Reality book similar to magic books, but it is not limited to the visual perception of the user but considers all human's five senses to provide additional information to the user [13]. Our digilog book consists of 10 pages. Each page includes pictures related to the book story. The important consideration for designing our digilog book was the correlation between virtual 3D models and figures illustrated on a page. As shown in [15], the real pages and the virtual objects should be harmonized for satisfying visual results. Thus, we designed the virtual contents to seamlessly appear from the real pictures as shown in Figure 10a. We made the proposed tracker run as a building block in Virtools [2], which is a commercial authoring tool for VR. It facilitates the authoring and the redistributions of the contents. Figure 10 shows the snapshots of the implemented digilog book. The frame-rate was still higher than 25 FPS with heavy contents (80MB).

6 Conclusions and Future Work

We presented a planar targets 6DOF tracker that handles more than 300 objects without loss of tracking performance. The proposed tracker used a multi-core programming for exploiting highly distinctive SIFT fea-

[2] http://www.3ds.com/products/3dvia/3dvia-virtools

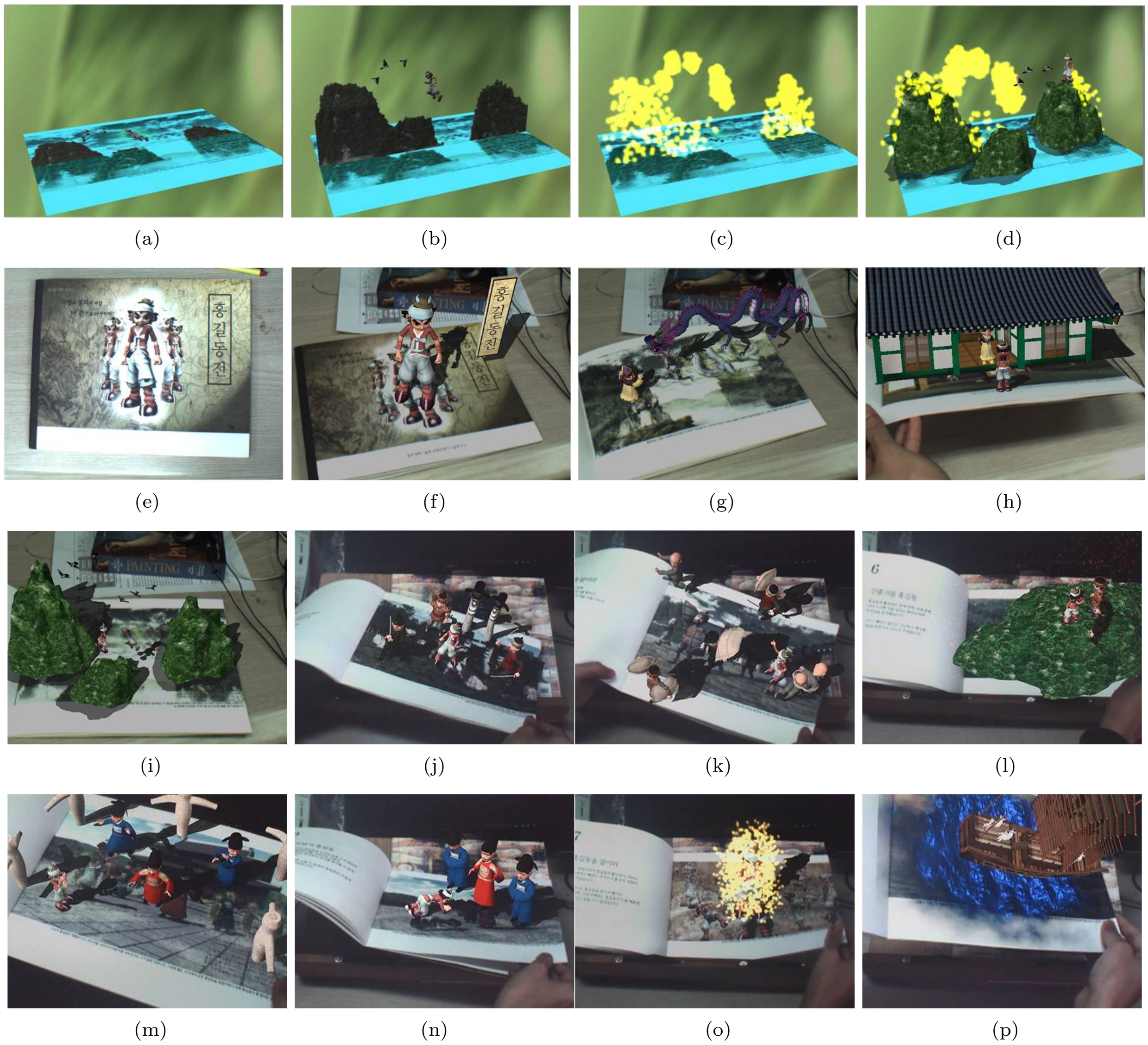

Fig. 10: Implemented digilog book (a)-(d) Moving virtual 3D content on the 4th page that produced from the commercial 3D tools (e) real book cover, (f)-(p) snapshots of the augmented contents at each page.

tures for a real-time application. The efficient vocabulary tree-based searching method was proposed to cover a large data set. We expect that the proposed framework can work with a thousand of targets with the help of vocabulary tree data structure. The approach can be easily extended to the multiple 3D objects tracking by replacing computing a camera pose from homography with a general one, such as PnP. It will be also possible to track multiple objects simultaneously with a small modification. We also showed that the proposed method was successfully used in an AR book application. In the future, SIFT feature can be replaced with light-weight one like [18] so that the tracker can work on mobile phones.

References

1. Bay, H., Ess, A., Tuytelaars, T., Vangool, L.: Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding **110**(3), 346 – 359 (2008)
2. Billinghurst, M., Kato, H., Poupyrev, I.: The magicbook - moving seamlessly between reality and virtuality. Computer Graphics and Applications, IEEE **21**(3), 6 – 8 (2001)
3. Fiala, M.: ARTag, a fiducial marker system using digital techniques. Conference on Computer Vision and Pattern Recognition **2**, 590–596 (2005)
4. Frauendorfer, F., Wu, C., Frahm, J.M., Pollefeys, M.: Visual word based location recognition in 3D

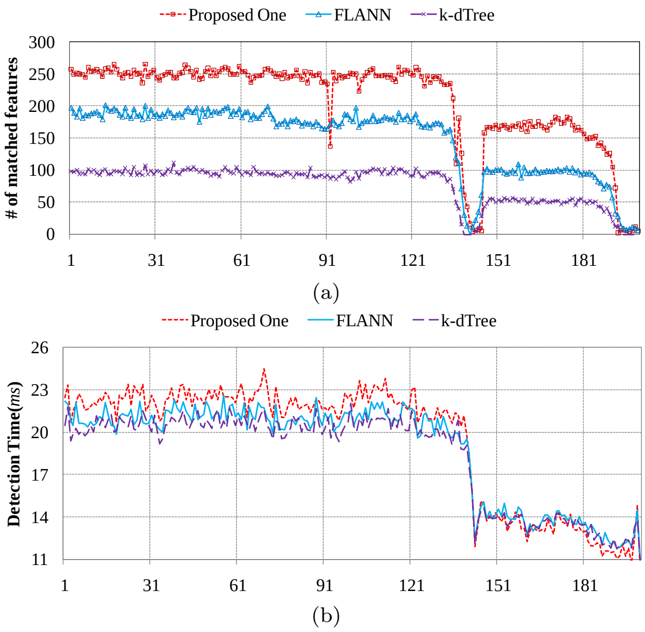

Fig. 7: Comparison results for the detection module. (a) The number of inliers in the sequence and (b) detection time.

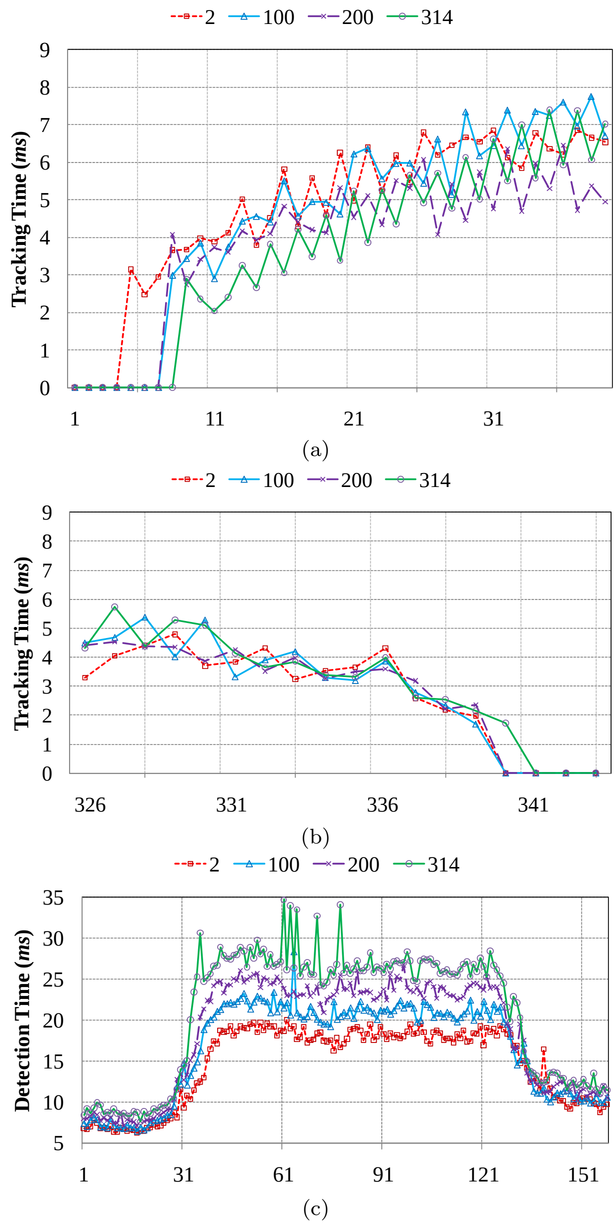

Fig. 9: Detection and tracking performance according to data size (Vocabulary trees with SIFT keypoints in 2, 100, 200 and 314 targets were constructed, respectively); (a) When the target is about to be detected, the 2-tree case showed the fastest detection time (b) When the target is about to disappear, the 314-tree case showed the latest response. (c) The detection module for 314-tree case had the longest time to recognize one target. However, we observed that it did not make significant difference compared to the 2-tree case in the overall tracking performance.

models using distance augmented weighting. In: 3D Data Processing, Visualization and Transmission (2008)

5. Ha, T., Lee, Y., Woo, W.: Digilog book for temple bell tolling experience based on interactive augmented reality with culture technology. In: Springer Journal of Virtual Reality (spVR) (accepted) (2009)
6. Kato, H., Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: 2nd IEEE and ACM International Workshop on Augmented Reality, pp. 85 – 94 (1999)
7. Lee, T., Hollerer, T.: Multithreaded hybrid feature tracking for markerless augmented reality. IEEE Transactions on Visualization and Computer Graphics **15**, 355 – 368 (2008)
8. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. In: Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 775 – 781 (2005)
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
10. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. International Conference on Compute Vision Theory and Applications pp. 331 – 340 (2009)
11. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: Conference on Computer Vision and Pattern Recognition, pp. 2161–2168 (2006)
12. Ozuysal, M., Calonder, M., Fua, P., Lepetit, V.: Fast keypoint recognition using random ferns. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(3), 448–461 (2010)

13. Park, J., Woo, W.: Multi-layer based authoring tool for digilog book. International Conference on Entertainment Computing, Lecture Notes in Computer Science **5709**, 234–239 (2009)
14. Park, Y., Lepetit, V., Woo, W.: Multiple 3D object tracking for augmented reality. In: International Symposium on Mixed and Augmented Reality, pp. 117–120 (2008)
15. Scherrer, C., Pilet, J., Fua, P., Lepetit, V.: The haunted book. In: International Symposium on Mixed and Augmented Reality, pp. 163–164 (2008)
16. Simon, G., Fitzgibbon, A., Zisserman, A.: Markerless tracking using planar structures in the scene. In: International Symposium on Mixed and Augmented Reality, pp. 137–146 (2000)
17. Taketa, N., Hayashi, K., Kato, H., Noshida, S.: Virtual pop-up book based on augmented reality. Symposium on Human Interface 2007, Held as Part of HCI International 2007, Lecture Notes in Computer Science pp. 475 – 484 (2007)
18. Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., Schmalstieg, D.: Pose tracking from natural features on mobile phones. In: International Symposium on Mixed and Augmented Reality, pp. 125–134 (2008)
19. Wagner, D., Schmalstieg, D.: Artoolkitplus for pose tracking on mobile devices. Proceedings of 12th Computer Vision Winter Workshop (CVWW'07) pp. 139 – 146 (2007)
20. Wagner, D., Schmalstieg, D., Bischof, H.: Multiple target detection and tracking with guaranteed framerates on mobile phones. In: International Symposium on Mixed and Augmented Reality, pp. 57–64 (2009)

Kiyoung Kim received his B.S. in the Department of Computer Science from the Chung-Ang University, Seoul, Korea, in 2003 and his M.S. in the Department of Information and Communications (DIC) from the Gwangju Institute of Science and Technology (GIST), Gwangju, Korea, in 2004. He has been a Ph.D. student in DIC, GIST since 2004. His research interests include 3D computer vision for augmented reality, 3D object tracking, computer graphics, and HCI.

Vincent Lepetit is a Senior Researcher at the Computer Vision Laboratory, EPFL. He received the engineering and master degrees in Computer Science from the ESIAL in 1996. He received the PhD degree in Computer Vision in 2001 from the University of Nancy, France, after working in the ISA INRIA team. He then joined the Virtual Reality Lab at EPFL (Swiss Federal Institute of Technology) as a post-doctoral 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 his B.S. in Electronics Engineering from Kyungpook National University in 1989 and his M.S. in Electronics and Electrical Engineering from POSTECH in 1991. In 1998, he received his Ph.D. in Electrical Engineering Systems from University of Southern California (USC). 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), where he is an Associate Professor in the Department of Information and Communications (DIC) and Director of Culture Technology Institute (CTI). His research interests include 3D computer vision and its applications including attentive AR and mediated reality, HCI, affective sensing and context-aware for ubiquitous computing.