Keyframe-based Modeling and Tracking of Multiple 3D Objects *

Kiyoung Kim† Vincent Lepetit‡ Woontack Woo§
GIST U-VR Lab. EPFL CVLab. GIST U-VR Lab.

ABSTRACT

We propose a real-time solution for modeling and tracking multiple 3D objects in unknown environments. Our contribution is two-fold: First, we show how to scale with the number of objects. This is done by combining recent techniques for image retrieval and online Structure from Motion, which can be run in parallel. As a result, tracking 40 objects in 3D can be done within 6 to 25 milliseconds per frame, even under difficult conditions for tracking. Second, we propose a method to let the user add new objects very quickly. The user simply has to select in an image a 2D region lying on the object. A 3D primitive is then fitted to the features within this region, and adjusted to create the object 3D model. In practice, this procedure takes less than a minute.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, Augmented, and Virtual Realities; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Shape

1 INTRODUCTION

Considerable progress has been made recently for camera tracking in unknown environments, in terms of reliability [7] and scalability with the size of the scene [4]. By contrast, tracking of 3D objects moving independently received much less attention, despite its importance for Human Computer Interfaces or Augmented Reality.

As Figure 1 shows, our contribution in this paper is two-fold. First, we show how to handle a large number of objects. For example, we can maintain real-time performances while simultaneously tracking 13 different objects, while continuously checking for objects from a database of 40 objects to appear. To do that, we built upon our approach developed in [6], which is able to manage a large database of planar objects but track only one object at a time. Object detection and feature tracking run in parallel: A 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. We show how to extend the latter to detect all the known objects present in an image to be able to track reliably several objects simultaneously.

The second part of our contribution is to show how to allow the user to add new objects very efficiently. In a way similar to [7], a second background thread maps the environment to track the camera and reconstruct the 3D locations of image features. When the user wants to define a new object, he simply has to select from a captured frame a region that lies on the object. We then fit automatically a 3D primitive to the 3D locations corresponding to the image features that belong to this region, to obtain the object 3D model. This procedure is fast and intuitive, and allows to create reliable 3D models and data required for tracking with a minimal effort. Our current implementation is limited to box-like primitives, but could easily be extended to other shapes.

In the remainder of the paper, we first address related work in Section 2. We then give an overview of our approach and detail it in Section 3. The experimental results are discussed in Section 4.

2 RELATED WORK

We review below previous work about multiple object tracking and interactive modeling, respectively.

2.1 3D Multiple Object Tracking

[10] proposed a real-time tracking algorithm for multiple 3D objects that combines detection and frame-to-frame tracking. The main drawback of the approach is that its complexity in terms of computation time and memory grows linearly with the number of objects, which makes it impractical when considering more than 10 objects. [16] optimizes a score evaluating the trade-off between detection and frame-to-frame tracking, as detection is more robust but requires more time, to dynamically asjust the tracking and detection loads at run-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. It can handle a remarkably large number of objects simultaneously visible, however it is still limited in the number of objects that can potentially become visible, in other words the number of known objects in the database.

Very recently, [11] developed an approach that can scale to a very large number of known objects, by relying on image retrieval techniques similar to the one we use [8]. In this paper, we show how to combine detection and tracking for multiple objects, so that these two parts of the approach can be run in parallel and communicate when needed. Thus, our method runs at more than 50Hz on a modern PC, while [11] reports about 8 Hz.

We also propose a method to let the user add new 3D objects very quickly, as discussed in the next subsection.

2.2 Online Interactive Modeling

Early works on online interactive modeling include [5] and [3]. [5] relies on a visual marker while we do not need to engineer the environment and use natural features for tracking and reconstruction. [3] introduced a special device that combines a camera and physical buttons for the interface. By contrast, we focus on using standard hardware.

Recently, [9] proposed a method to interactively build complex 3D models of real objects. The system uses AR for user guidance through the reconstruction process, and does not really focus on tracking the object afterward, as we do.

The *in situ* modeling in [12] is closer to our approach as the user can define the vertices of the object mesh while moving the camera. We show here that we can combine automated reconstruction and interactive modeling by fitting 3D primitives to the reconstructed 3D locations. This makes the task much easier for the user.

Maybe closer to our own work is [15], which combines VideoTrace [14], an off-line reconstruction method from videos, with PTAM [7] for interactive modelling. However this approach focuses on modelling the scene, while we can build models of independently moving objects.

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

†e-mail: kkim@gist.ac.kr
‡e-mail:vincent.lepetit@epfl.ch
§e-mail:wwoo@gist.ac.kr

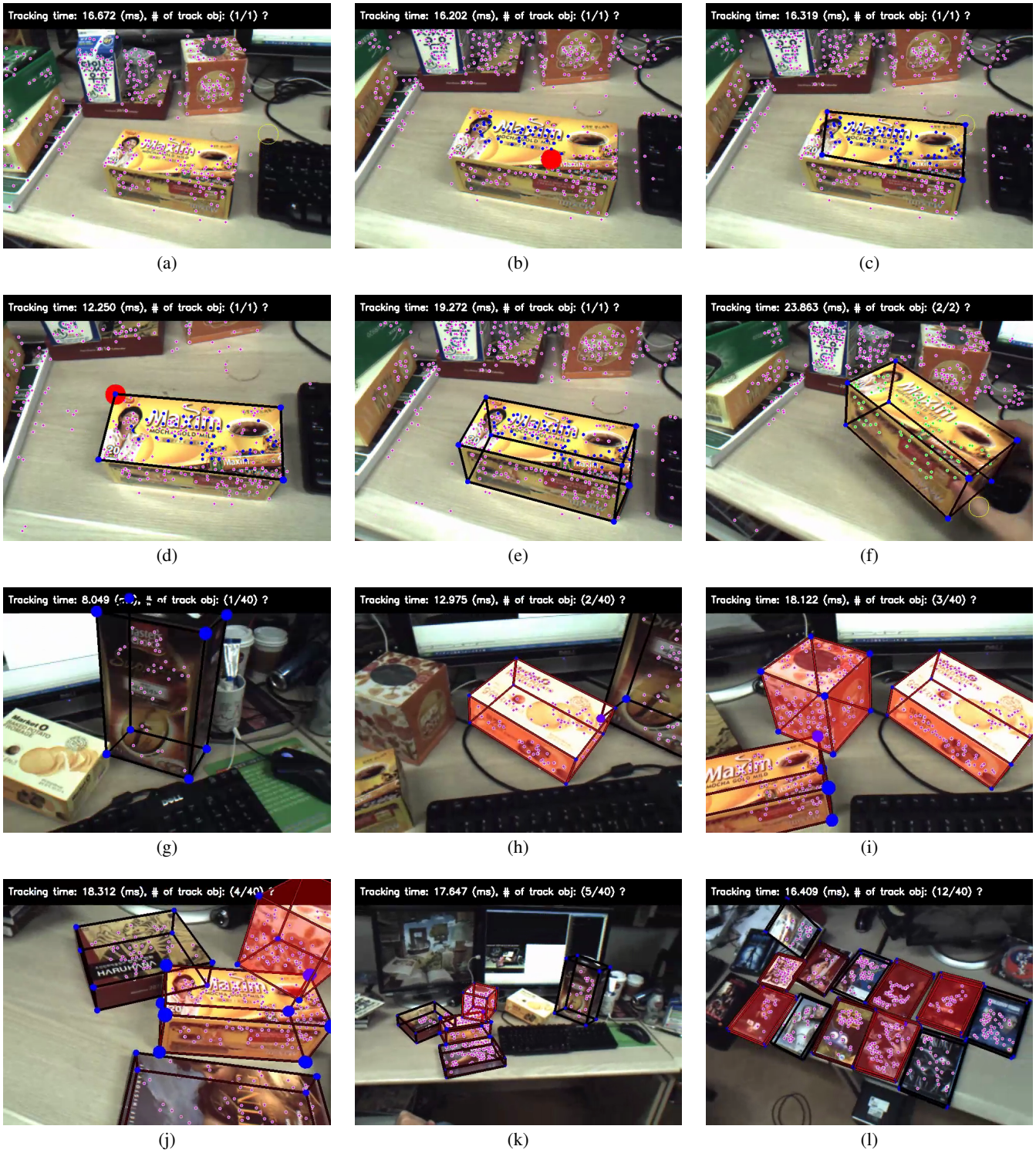


Figure 1: Overview of our approach. In the top line (A/B), (A): the total number of the detected objects in the current image, and (B): the number of the registered objects in database. (a) Our system continuously reconstructs the environment and tracks the camera from the video stream. (b) To add a new object, the user starts by selecting some features on the object using a brush-like tool. (c) The system then fits a planar facet to the features that can be adjusted (d) and extended to a 3D box by the user (e). (f) The object can then be moved independently from the rest of the scene. (g-k) Known objects are recognized when they become visible, and are tracked independently from each other. (l) Our system can handle many objects in real-time.

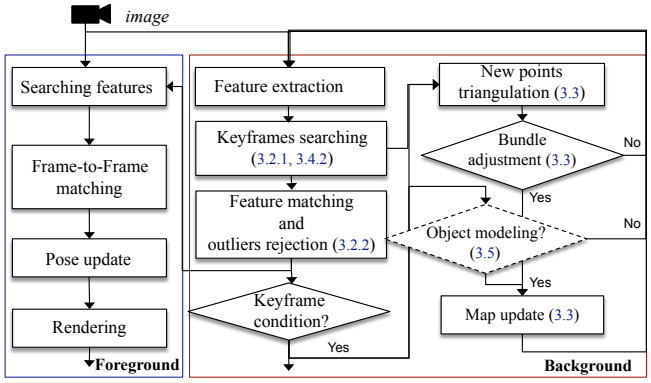


Figure 2: Flowchart of the proposed tracking, mapping and modeling.

3 Tracking, Mapping and Modeling

3.1 Overview

Our general approach is to track the camera and the known objects and reconstruct the 3D environment continuously. At any time, the user can interactively define new objects from the captured images and the environment reconstruction.

As shown in Figure 2, our system is made of three different processes that run in parallel for efficiency. In a way similar to PTAM [7], the foreground process mostly computes the pose of the camera and the objects from frame-to-frame matches, and one background process takes care of the online reconstruction. An additional background process matches the input images with keyframes of the environment and the objects. This makes tracking more robust than the frame-to-frame tracking alone and allows to recognize known objects when they become visible and start tracking them.

The user initiates the object modeling process by selecting keypoints from the reconstructed environment that lie on a planar patch on the object to model.

We detail below these different aspects of our system. We first focus on camera tracking and environment reconstruction, and then show how to extend them to model and track multiple objects.

3.2 Camera Tracking

Our approach to camera tracking is very similar to the one we developed for digilog books, or magic books, in [6]. As mentioned above, it is made of two processes, a fast one, the "tracking module", that relies on frame-to-frame tracking and runs in the foreground, and a slower one, the "detection module", that tries to match the input frames with keyframes of the environment and of the known objects.

We quickly describe these two modules below. More details can be found in [6].

3.2.1 Detection Module

The environment reconstruction process, which will be detailed in Section 3.3, generates *keyframes* that are images containing keypoints that have been reconstructed in 3D. The detection module quickly retrieves the keyframes similar to the input frame, and matches their keypoints with those extracted in the input image.

For a given input image, we first extract SIFT keypoints from it and run the algorithm described in [8] for image retrieval. This algorithm is based on a vocabulary tree to identify the keypoints and look-up tables to quickly retrieve similar images. 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 keyframes.

To actually match the keypoints between the keyframes and the input frame, we use kd-trees. Each keyframe has an associated kd-tree, and the kd-tree for the input frame is computed online. We finally use RANSAC to robustly compute the camera rotation and translation from these matches.

3.2.2 Tracking Module

When the detection module finished to match the input frame with a keyframe, this information can be used by the tracking module. Unfortunately, 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 [13] for example. In practice we use 16×16 correlation windows.

We can then proceed to track feature points over consecutive frames. The camera pose is estimated using their 3D locations as recovered by the environment reconstruction process, described below.

3.3 Environment Reconstruction

Like in PTAM, the initial map is obtained from two views. The map is then extended by first matching an input frame and the corresponding keyframe as recovered by the process explained in Section 3.2.1.

The input frame is added as a keyframe and contributes to the environment reconstruction if it passes several conditions: It must exhibit a sufficient large number of new keypoints, the reprojection error must be sufficient low, and the corresponding camera center must be distant enough from the camera center of the corresponding keyframe to allow an accurate reconstruction. If it does, we reconstruct the 3D locations of the new keypoints matched with the corresponding keyframe but do not belong to the map yet. Finally, we perform a local bundle adjustment using the 10 keyframes closest to the input frame in a background process.

Input: SIFT features S extracted from the input camera image
Output: Inliers of each object

K: set of keyframes returned from vocabulary tree ;

K $\leftarrow$ queryToVocabularyTree(S) ;

Keep only the first min(# of objects $\times$ 5, # of total keyframes) best keyframes in **K**;

```
foreach keyframe K ∈ K do
    O ← correspondingObject(K) ;
    if O is not currently tracked then
        if no good keyframe for O has been found yet then
            [inliers] ← robustMatch(O, S) ;
            if # of [inliers] > T_val then
                send [inliers] to foreground tracking thread ;
            end
        end
    end
end
```

Algorithm 1: Pseudo-code for multiple objects detection.

3.4 Multiple Object Tracking

We now explain how we extend the camera tracking process to track multiple objects, and describe how the user can quickly define new objects.

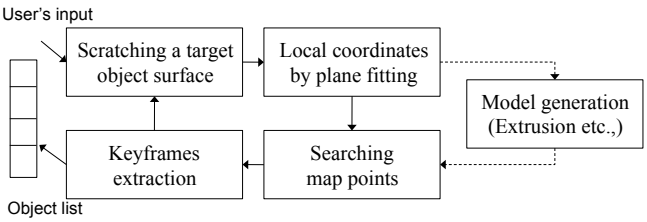


Figure 3: Flowchart of the proposed modeling procedure.

3.4.1 Object Tracking Algorithm

To each object corresponds a set of keyframes similar to those of the environment as used in Section 3.2. The only difference is that these keyframes contain only the keypoints that lie on the corresponding objects. These keyframes are created automatically during the interactive object modeling as will be explained in Section 3.5. They are referenced in the detection module by a second vocabulary tree. We could use the same vocabulary tree as for the keyframes for the environment but we noticed that keeping the two types of keyframes results in better performances.

The pseudo-code for detecting newly visible objects is given in Algorithm 1. For a given input frame, the detection module returns a list of keyframes sorted by similarity score. We consider at least one keyframe per object in the database: If one object exhibits much more keypoints than another one, it tends to artificially get better scores, and appears more often in the best keyframes. We then try to robustly match each keyframe from this list with the input image, using RANSAC to compute the object pose. If the number of inliers is sufficient large, the object is considered visible in the input image, and its keypoints are sent to the foreground process and tracked frame-by-frame.

3.5 Interactive Object Modeling

We finally describe here how the user can define new objects easily.

3.5.1 User Interface

Figure 3 shows the flowchart of the proposed modeling procedure. The modeling process is activated by the user's input. It eventually results with a set of keyframes corresponding to the modeled object. These keyframes will be used to track the object as described in the previous section.

In the first step, for selecting a target object area, we use a circular 2D brush pointer to mark features from images. The user can use a 2D brush-like tool to mark directly in the input frames some keypoints lying on the objects. For convenience, the brush radius can be changed adaptively with a mouse wheel movement in order to pick keypoints one by one or entire set at a time.

Once the keypoints are selected, we fit a plane to the corresponding 3D points. Four virtual corner points are displayed and the user can adjust their positions on the plane without pausing the tracker. The 2D surface can finally be extended in a direction orthogonal to the 3D plane to define the object 3D shape.

3.5.2 Keyframes Selection

To create the set of keyframes for the modeled object, we consider the keyframes already used to track the environment. Using the 3D locations of the keypoints that belong to the object, it is easy to find the keyframes where these keypoints are also visible. We retain the keyframes with the largest ratios between the number of keypoints lying on the object and the total number of keypoints extracted in the keyframe.

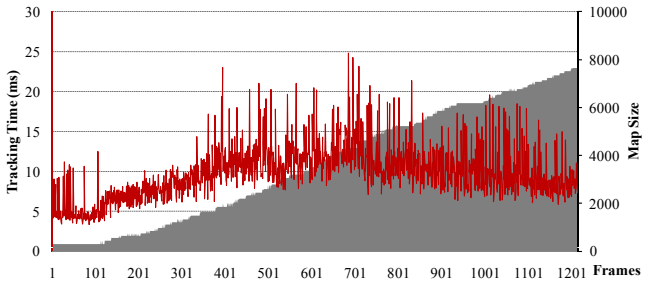
Only the keypoints that lie on the object are kept. The keyframes are finally added to the vocabulary tree of the detection module, and the kd-tree of each keyframe is computed. This is carried out without stopping tracking.

4 Experimental Results

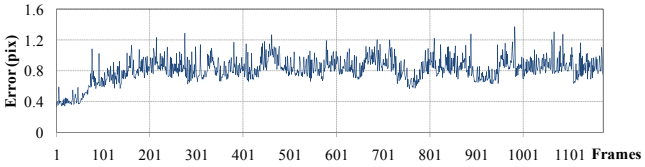
In all our experiments, we used a PC with an Intel 2.95 GHz Quad-core CPU and a NVidia GTX285 graphic card, and a USB type FireflyMV camera [2] with a 3.5 mm lens. We used two vocabulary trees (one for the environment, one for the objects) each made of 6 levels and 10 branches per level, and the implementation from [1]. We also used SIFTGPU [17] to speed up feature extraction. The other parts of the system were implementing using OpenCV [1].

4.1 Real-Time Tracking and Mapping Performance

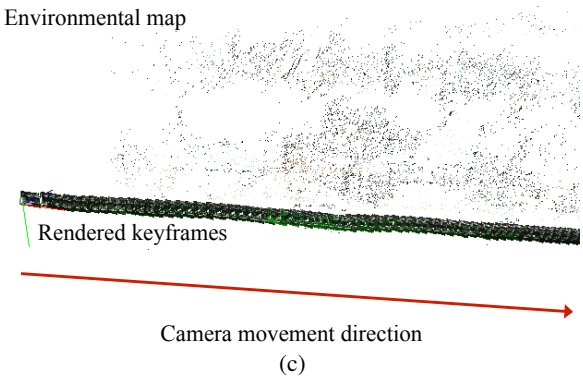
Figure 4 shows the results of the proposed tracker running in a small desktop environment. For this experiment, we translated the camera roughly in a direction orthogonal to its line of sight, and parallel to the ground plane. Then we measure the tracking time and the reprojection error over 1000 frames. The results are shown in Figure 4. Tracking time always remained between 5 and 25 ms, and the reprojection error was maintained under 2 pixels, even when the map contains almost 8000 points. The keyframes and map points are rendered in 3D in Figure 4(c), which shows that the camera trajectory is as expected, at least qualitatively.



(a)



(b)



(c)

Figure 4: Tracking and mapping results for a camera in translation. (a) Tracking time in milliseconds and the number of features in the map, and (b) Reprojection error in pixels over 1,000 frames. The x-axis corresponds to the frame index. (c) The map points and the camera for the keyframes.

[1] http://www.cvg.ethz.ch/people/postgraduates/fraundof/vocsearch

Table 1 details the average times for each module. The real-time performances depends on the number of visible objects and features tracked in the current frame, but the frame-rate of the tracking module is always higher than the video capture frame rate.

Table 1: Average Times for the Different Modules Running on (1): Foreground and (2),(3): Background Threads.

	Process module	Time (*ms*)
(1)	Searching features	1.876
	Frame-to-Frame matching	3.432
	Pose update per object	2.128
(2)	Keyframe searching (vocabulary tree)	11.061
	Feature matching (kd-tree)	11.848
	Outliers rejection	2.386
	Guided matching	1.284
(3)	Keyframe insertion (vocabulary tree, kd-tree)	16.215 + 21.092

4.2 3D Object Modeling and Multiple Objects Tracking Results

Figure 5 shows an example of modeling and tracking a single box by using the proposed method. In this case, a total of 32 object-keyframes were created. With the help of the strong distinctiveness of SIFT features we could track the target box from previously unseen viewpoints.

To evaluate the scalability of the proposed method with the number of objects, we modeled 40 different objects —some of them are shown in Figure 6(a)— for a total of 444 keyframes and 24,568 map points. Figure 6(b) shows the number of keyframes and collected map points for 10 of the objects.

Figure 6(c) shows the evolution of the computation time required for tracking together with the number of objects visible in the image. Up to 13 objects were visible, and the computation time varied from 6 ms to 25 ms. The computation time evolves roughly linearly with the number of *visible* objects, while the number of *known* objects has only a very limited influence.

REFERENCES

[1] Open computer vision library. *http://sourceforge.net/projects/opencvlibrary/*, 2009.

[2] Point grey research. *http://www.ptgrey.com/*, 2009.

[3] P. Bunnun and W. Mayol-Cuevas. OutlinAR: An Assisted Interactive Model Building System with Reduced Computational Effort. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2008.

[4] R. Castle, G. Klein, and D. W. Murray. Video-Rate Localization in Multiple Maps for Wearable Augmented Reality. In *IEEE International Symposium on Wearable Computers*, pages 15–22, 2008.

[5] R. Freeman and A. Steed. Interactive Modelling and Tracking for Mixed and Augmented Reality. In *Virtual Reality Software and Technology*, 2006.

[6] K. Kim, V. Lepetit, and W. Woo. Scalable Real-Time Planar Targets Tracking for Digilog Books. *The Visual Computer*, 26(6-8):1145–1154, 2010.

[7] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2007.

[8] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2006.

[9] Q. Pan, G. Reitmayr, and T. Drummond. Interactive Model Reconstruction with User Guidance. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2009.

[10] Y. Park, V. Lepetit, and W. Woo. Multiple 3D Object Tracking for Augmented Reality. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2008.

[11] J. Pilet and H. Saito. Virtually Augmenting Hundreds of Real Pictures: An Approach Based on Learning, Retrieval, and Tracking. In *Virtual Reality*, 2010.

[12] G. Simon. In-Situ 3D Sketching Using a Video Camera as an Interaction and Tracking Device. In *Eurographics*, 2010.

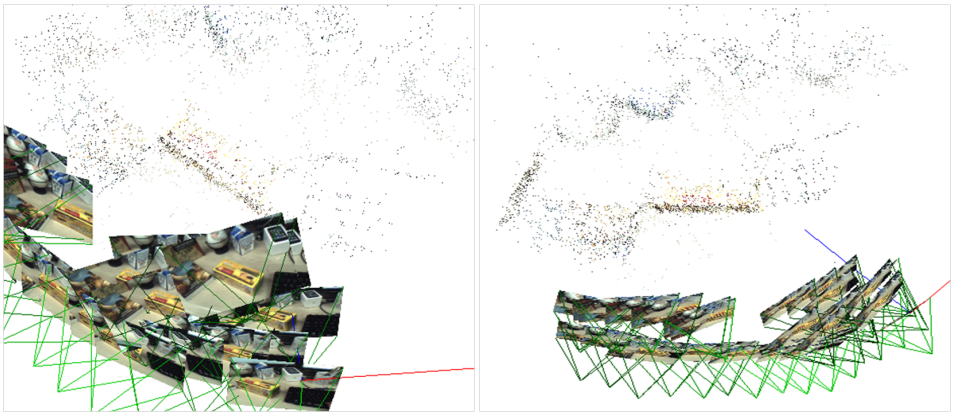
[13] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless Tracking Using Planar Structures in the Scene. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 120–128, October 2000.

[14] A. Van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. VideoTrace: Rapid Interactive Scene Modelling from Video. In *ACM SIGGRAPH*, 2007.

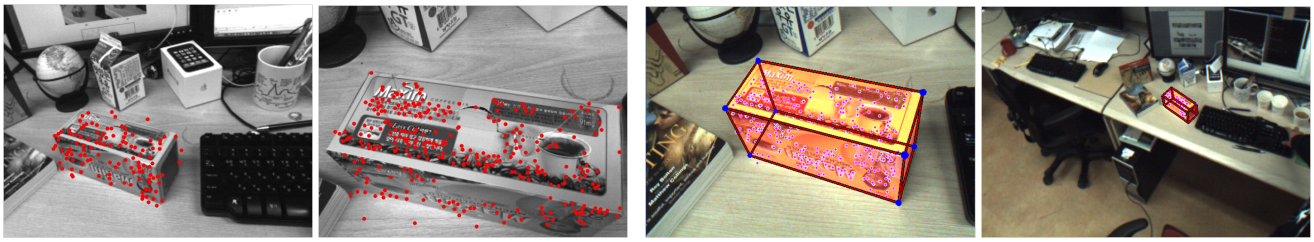
[15] A. Van den Hengel, R. Hill, B. Ward, and A. Dick. In Situ Image-Based Modeling. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2009.

[16] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple Target Detection and Tracking with Guaranteed Framerates on Mobile Phones. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2009.

[17] C. Wu. SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT), 2007.



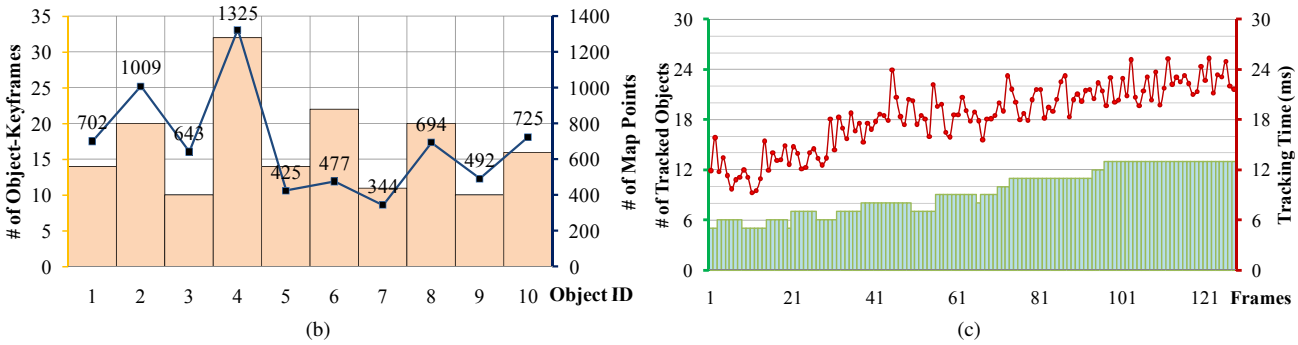
(a)



(b) (c)

Figure 5: Object modeling and tracking using the proposed method: (a) Generated map and tracker keyframes during the modeling process described in Figure 1(a)-(f) seen from two different viewpoints, (b) two of the generated object-keyframes and their object-keypoints represented as red dots, and (c) two snapshots of the target object tracking results.

(a)



(b) (c)

Figure 6: Evaluating the scalability in terms of the number of objects: (a) 10 of the 40 objects used for the experiment. (b) Number of keyframes and number of map points for each object of these 10 objects. (c) The prototype could detect and track potentially visible 40 objects, for a total of 444 keyframes and 24,568 map points. The computation required for tracking reached 25 ms when 13 objects were visible and tracked simultaneously. The computation time evolves roughly linearly with the number of *visible* objects, while the number of *known* objects has only a very limited influence.