Marker-less Tracking for Multi-layer Authoring in AR Books

Kiyoung Kim, Jonghee Park, and Woontack Woo *

GIST U-VR Lab.
500-712, Gwangju, S. Korea
{kkim, jpark, wwoo}@gist.ac.kr

Abstract. An Augmented Reality (AR) book is an application that applies AR technologies to physical books for providing a new experience to users. In this paper, we propose a new marker-less tracking method for the AR book. The main goal of the tracker is not only to recognize many pages, but also to compute 6 DOF camera pose. As a result, we can augment different virtual contents according to the corresponding page. For this purpose, we use a multi-core programming approach that separates the page recognition module from the tracking module. In the page recognition module, highly distinctive Scale Invariant Features Transform (SIFT) features are used. In the tracking module, a coarse-to-fine approach is exploited for fast frame-to-frame matching. Our tracker provides more than 30 frames per second. In addition to the tracker, we explain multi-layer based data structure for maintaining the AR book. A GUI-based authoring tool is also shown to validate feasibility of the tracker and data structures. The proposed algorithm would be helpful to create various AR applications that require multiple planes tracking.

Keywords: augmented reality, marker-less tracking, layer authoring, page recognition, AR book, SIFT

1 Introduction

Camera tracking plays an important role in the implementation of Augmented Reality (AR) applications. Generally, the purpose of camera tracking in AR is to compute a relative camera pose, represented in a rotation and a translation matrix, with respect to the local coordinates of a tracked object. Then, the camera pose is used to augment virtual contents with the projection matrix obtained from camera intrinsic parameters. For AR book applications it is mandatory to recognize which page is visible and to compute a camera pose in real time. In particular, AR book applications requires a robust page number extraction because it allows developers to map different virtual contents onto the corresponding page of a book. The stability and accuracy of camera poses are also

* This research was supported by the CTI development project of KOCCA, MCST in S.Korea.

important issues as well as the speed of tracking. An unstable and slow tracker may cause users to lose immersion to the AR books. Additionally, a real-time page recognition method is also requires for the AR books. With AR books, users can not only read the story written in the traditional way, but also view and manipulate 3D models with the help of a good camera tracker. Moreover, AR books offer improved user experiences by providing new applications which were not available with traditional books, such as 3D virtual games or storytelling.

Many AR books with camera trackers have been developed [1–3]. In the early stage, fiducial markers were used to get a camera pose relative to the markers [1]. The fiducial markers are convenient for identifying each page and computing camera poses. Additionally, attaching the markers on the physical tools provides a way of interaction which allows direct manipulation of augmented objects. However, the markers distract users' concentrations and are sensitive to occlusions. Recent AR books adopt tracking by detection-based marker-less tracking methods [2][3]. The marker-less methods are robust to occlusions and do not require any distinctive markers onto pages. However, most of the marker-less tracking methods require high computational costs and are not much faster than marker-based methods. Moreover, the fast method, like [4] requires a lot of time and computer memory in training phase so that it takes long time for authoring the contents. A hybrid method which uses markers for page recognition and randomized trees for camera tracking [4] was proposed in order to support multiple pages [5].

In this paper, we propose a new marker-less tracking method which supports real time page recognition and fast camera tracking. We use a multi-core programming approach which separates a page recognition from a camera tracking module in order to improve the performance without losing accuracy and robustness of the tracker. Our method adopts highly distinctive Scale Invariant Features Transform (SIFT) features [6] for the page recognition. We extract SIFT features from each page and save them as a template in the offline process. When an input image comes into the tracker, the tracker compares the SIFT features of the input image and the saved SIFT features template. The processing time for the comparison process is varied according to the total number of features in each page. We proposed an efficient voting-based method to filter out irrelevant pages to reduce the processing time for the recognition. In a camera tracking phase, we perform frame-to-frame matching based on a coarse-to-fine approach with FAST corners [7]. In addition, we explain a multi-layer, a set of subregions of a page, data structure for maintaining the AR book. The Graphical User Interface (GUI) based authoring tool is shown to validate feasibility of the proposed tracker and the multi-layer based data structure.

2 Related Work and Background

2.1 Mark-less tracking methods

There have been recent developments on marker-less tracking. We categorize the marker-less tracking into two groups according to its prior conditions: The first one is detection-based tracking (tracking by detection) methods [4, 8–10]; the second one is SLAM-based methods [11–14]. While the detection methods have been used mainly in pre-defined object tracking, the SLAM-based methods have been used in unknown environments. Thus, for AR book applications, detection-based tracking methods are more appropriate than SLAM-based methods because we already know which page are included in the AR book. The important problem of the detection-based methods is the processing time for the feature recognition. Trackers with SIFT [6] or SURF [15] was not enough for real time AR applications due to heavy computational costs. To provide real time recognition, the randomized tree (RT) was adopted [4, 10]. However, the RT generation takes around 1 minute per one page. And it is not easy to utilize RT method in a multiple pages recognition because of the large amount of memory for keeping RT structures. It is not desirable characteristic for AR book designers or developers.

Our method overcomes the limitations in training time and the camera frame rate by adopting a multi-core programming approach. The proposed method differs from [14, 16] in which we maintain two different SIFT and FAST features for the page recognition and frame-to-frame matching. We get the benefits of both, SIFT features and FAST corners, so that the tracking is done in less than 10ms. Only orthogonal images of the pages are required to start tracking and the preparation (training) time takes less than one second per page. Thus, the proposed method is efficient to use in AR authoring.

2.2 Background

The pinhole camera model is adopted as the camera model for the proposed tracking algorithm. A 3D point $\mathbf{X} = \begin{pmatrix} X, Y, Z, W \end{pmatrix}^T$ in homogeneous coordinates is projected onto $\mathbf{x} = \begin{pmatrix} u, v, w \end{pmatrix}^T$ using Eq. 1:

$$\mathbf{x} = K \begin{pmatrix} R \, t \end{pmatrix} \mathbf{X} \tag{1}$$

where R and t is a rotation and a translation matrix, respectively. K is a camera intrinsic matrix consisting of focal length, principal point and so on. Let us assume that a page of a book is planar. In this case, the Z-value of 3D point $\mathbf{X}$ becomes zero. Thus, the relation between 3D points $\mathbf{X}$ and its projected point $\mathbf{x}$ is represented as a plane transformation, Homography H as shown in Eq. 2:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \underbrace{K \begin{pmatrix} R_{3\times 2} \; t_{3\times 1} \end{pmatrix}}_{H_{3\times 3}} \underbrace{\begin{pmatrix} X \\ Y \\ W \end{pmatrix}}_{\mathbf{M}} \tag{2}$$

In many AR book applications, it has been assumed that model points $\mathbf{M}$ is known as a priori with the assumption of $W = 1$ without loss of generality. The model points are usually obtained from an orthogonal image of a page by applying existing feature detectors [6, 15]. Thus, we should know the correspondences of model points and their observed points in the input image to compute H. Then, given K, the camera pose $\left(R\,t\right)$ is decomposed from H by using a rotation property [17].

The goal of the page recognition is to provide adequate model points $\mathbf{M}$ when we compute H. It is very inefficient to use all of model points of all the pages because a large number of model points require considerable processing time to recognize each page as well as computational memory. The proposed method reduces the recognition time by comparing the input image with the candidate pages which have high probability.

3 Proposed Approach

3.1 Preparation and Overall Procedure

We take an orthogonal image ($\mathcal{O}$) of a page as a template. Then, we extract SIFT features ($\mathbf{S}$) from the orthogonal image. And also, we apply FAST corner ($\mathbf{F}$) detector to get additional points on the page. As a result, we have $\mathcal{O}_i, \mathbf{S}_i, \mathbf{F}_i$ of i_{th} page. In addition, we generate pyramid images ($\mathcal{L}_{i,s}$) from $\mathcal{O}_i$ where s is a pyramid level from 0 to 4. We apply the FAST corner detection for each level and keep the output points of each level for frame-to-frame matching. The process is done in offline. Thus, it does not affect the performance of the tracker.

As shown in Figure 1, the proposed marker-less tracker uses two threads. The main thread where the camera tracking is carried out via a frame-to-frame matching determines an overall frame rate. The other module, the page recognition process, is repeatedly performed in the background thread, which is relatively slower than camera tracking in the main thread. We explain details of two modules in Section 3.2 and Section 3.3, respectively.

3.2 Real-time Page Recognition

In this section, we explain the method that determines a visible page. We adopt SIFT features for the page recognition. The SIFT is highly distinctive so that it can support multiple pages. In addition, it does not require a lot of time to process when we make tracking data compared to other method, like [4], which is an efficient property for authoring of many pages. The output of the method are page ID (*pid*) and the Homography (H) induced from matching results between the input image and $\mathcal{O}_{pid}$. The problem is, specifically, to find out parameters by comparing SIFT features of input image (S_{input}) with the saved features in the preparation step (3.1) which satisfy Eq. 3.

$$E^* = \underset{H,pid}{\operatorname{argmin}} \|\mathcal{S}_{input} - HS_{pid}\| \tag{3}$$

The goal of our method is to reduce the processing time when an AR book has a large number of registered pages. The first step to achieve the goal is evaluating

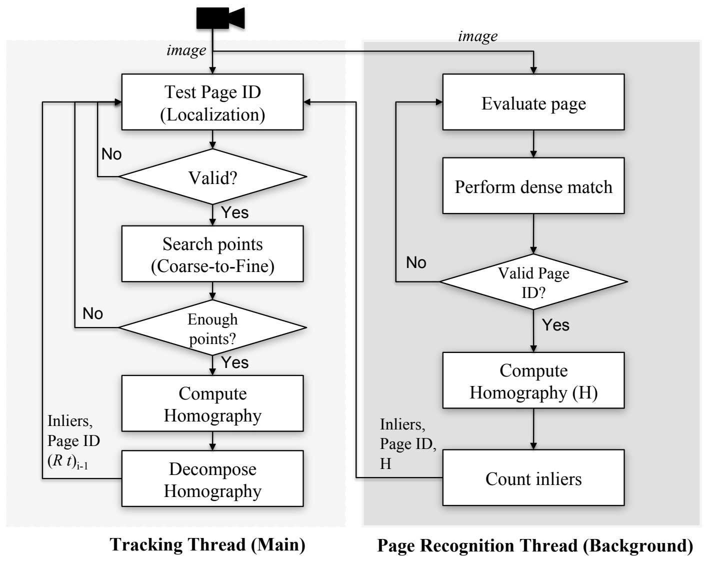


Fig. 1. Overall flowchart of the proposed tracker using two threads

all pages at every frame with given an input image. We use a voting scheme in the first step. We compute the defined scores at each page, and then sort the pages in a high score order. Then, the dense matching process is performed step by step from the highest-score-page to the lowest-score-page. The score function of k_{th} page is shown in Eq. 4:

$$\textbf{Score}(k) = -1 \times \{\omega_1 E_{DIST}(k) + \omega_2 E_{SSD}(k)\} \tag{4}$$

where E_{DIST} and E_{SSD} evaluate a sequential and a similarity constraint, respectively. ω_1 and ω_2 are weighting factors between two evaluation functions. E_{DIST} score measures how much the page is far from the lastly recognized (or currently viewing) page (pid^*) as shown in Eq. 5:

$$E_{DIST}(k) = \|pid^* - k\| \tag{5}$$

Eq. 5 allows to begin with the closest page matching. E_{SSD} score measures how much the input image is similar with the reference pages. We generate low level t pyramid image ($\mathcal{L}_{input,t}$) of the input image. And we perform Sum of Squares Difference (SSD) with $\mathcal{L}_{input,t}$ and each page pyramid image $\mathcal{L}_{k,t}$.

$$E_{SSD}(k) = \textbf{SSD}\left(\mathcal{L}_{k,t}, \mathcal{L}_{input,t}\right) \tag{6}$$

When the image resolution is 640×480 pixels and $t = 4$, the image size of $\mathcal{L}_{k,4}$ becomes 40×30 pixels. To reduce computational time, we limit the number of candidate pages. If we have the set of page scores $C^* = \{c_i > c_j > c_k, ...\}$, we compute the ratio values between adjacent pages, for example, $r(i,j) = \frac{c_i}{c_j}$. If the ratio is drastically changed, we ignore the consequent pages.

Based on the candidate pages from the process mentioned earlier, we perform dense matching process in the second stage. In the dense matching process, we compare $\mathcal{S}_{input}$ with SIFT features of candidates pages sequentially. We compute Homography and count inliers during the dense matching process. If the number of inliers satisfies our condition, then the system recognize the visible page as the recognized one. After the visible page is identified, we compute Homography and count inliers again. Finally, the page ID, Homography, and inliers information are sent to the main thread. For speeding up matching process, k-d tree can be used. Therefore, we can easily get the page ID without comparing all features in all pages.

3.3 Marker-less Tracking

In this section, we explain the tracking thread illustrated in Figure 1. The main thread is used for frame-to-frame matching, which matches points from two sequential images. The frame-to-frame matching enables a fast and an easy way to handle points because the movement of points between each frame is short. The tracking thread has i) localization and ii) frame-to-frame matching modules.

The localization is to find the current locations of the saved FAST corners with given Homography (H) obtained from the page recognition process. From the localization process, we find out which points are visible and valid for tracking. This is tested whenever the recognition module passes the results (page ID, H, and inliers) to the main thread. We need to search the corresponding points again because H often yields inaccurate results. For this purpose, we randomly select few sample points and warp patches of them by H. Then, SSD is performed for the selected points. After we find the corresponding points, we perform the guided matching within very narrow area. Then, we accept only if the reprojection error (R_ϵ) and the number of inliers satisfy the below conditions in the validation test.

$$R_\epsilon < T_{init} \ and \ \frac{\text{\# of Inliers}}{\text{Total \# of Points}} > T_{inliers} \tag{7}$$

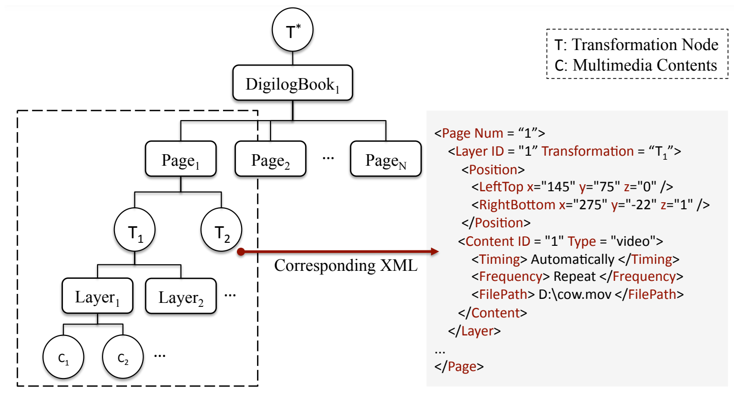
From the experiments, we found the tracker works well when T_{init} is 2.0 pixels and $T_{inliers}$ is 0.1. The localization enables recovery of the camera pose when tracking fails.

The frame-to-frame matching starts with the Homography (H_{prev}) of a previous frame and matched points. While the localization is performed only when the page recognition results are available, the frame matching is called at every new frame. First of all, we project high-scored 50 points of $\mathbf{F}_{pid}$ by H_{prev} and search the corresponding points to what within a circular area with radius ρ_b. Then, we compute Homography H_c with the coarse matching results. We project many points based on H_c and search again within a reduced circular area with radius $\frac{\rho_b}{2}$. Finally, we obtain the refined Homography $H_{final}$with many matches and warp the patch at each step with given Homographies. To improve the speed of the tracker, only if the initial error is small, we use the matched patches in the current image without warping. The final step is to decompose the H_{final} into rotation and translation matrix [17].

3.4 Multi-layer Authoring

One of obstacles in building AR book applications with a marker-less tracker is positioning virtual contents. In particular, content-aware augmentation, which augments virtual contents in accordance with real figures, texts, or user-defined regions in the page, cannot be realized without analyzing a page layout and layers. In this section, we address a multi-layer authoring scheme to realize a content-aware augmentation efficiently.

We define a page layout $\mathcal{R}_i$ as an arrangement and a style of virtual contents on the page. $\mathcal{R}_i$ of the page has multiple sub-layers $\{r_1, r_2, ..., \}$ which are defined by users. We use a scene graph structure for representing the multiple layers. Users define a sub-layer by dragging or picking a virtual point and allocate contents to the sub-layer. Currently we provide users pictures, text, sound, videos, and 3D CG models as virtual contents. Each layer is defined with several virtual points by the users and the defined layers are saved. Figure 2 shows a scene graph representation of the AR book and its Extensible Markup Language (XML) representation. The AR book consists of multiple pages. Additionally,



```xml
<Page Num = “1”>
  <Layer ID = "1” Transformation = “T1”>
    <Position>
      <LeftTop x="145" y="75" z="0" />
      <RightBottom x="275" y="-22" z="1" />
    </Position>
   <Content ID = "1" Type = "video">
      <Timing> Automatically </Timing>
      <Frequency> Repeat </Frequency>
      <FilePath> D:\cow.mov </FilePath>
   </Content>
  </Layer>
…
</Page>
```

Fig. 2. Illustration of page data structure and its example of XML representation

each page of the AR book is able to include multiple sub-layers which have areas of various contents. Each sub-layer can have multiple contents. When users create a sub-layer or area using an input device, the transformation that includes rotation and translation values is stored in the parent node of the layer.

Figure 3 shows the overall procedure of a multi-layer based authoring. When users make an AR book, a multi-layer authoring system constructs a scene graph of the AR book for rendering virtual contents in an AR environment. The generated multi-layer of the page is stored in XML format to facilitate the modification of the layers later in the layer management module. The AR book Viewer loads XML file according to a visible page ID in online mode. After that, the AR book viewer parses the XML file. After parsing page layers information, contents are augmented on top of the page.

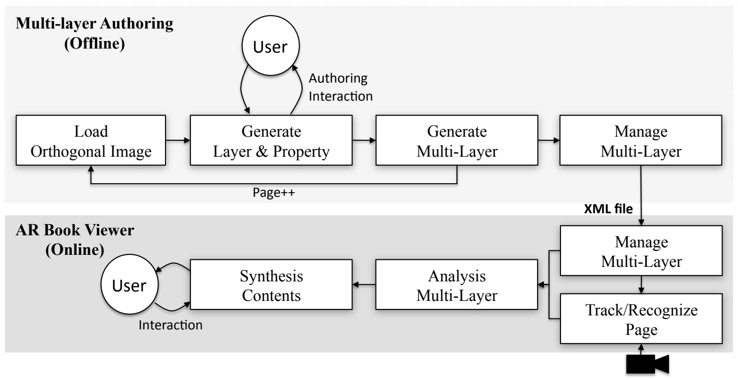


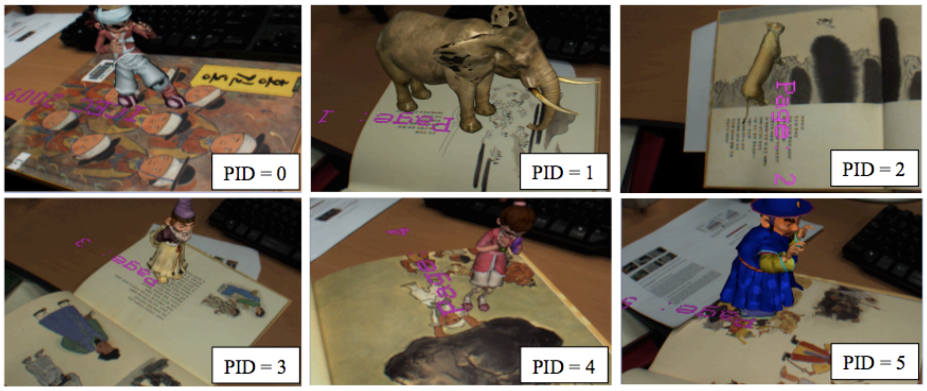
Fig. 3. Procedure for the multi-layer authoring

4 Implementation and Results

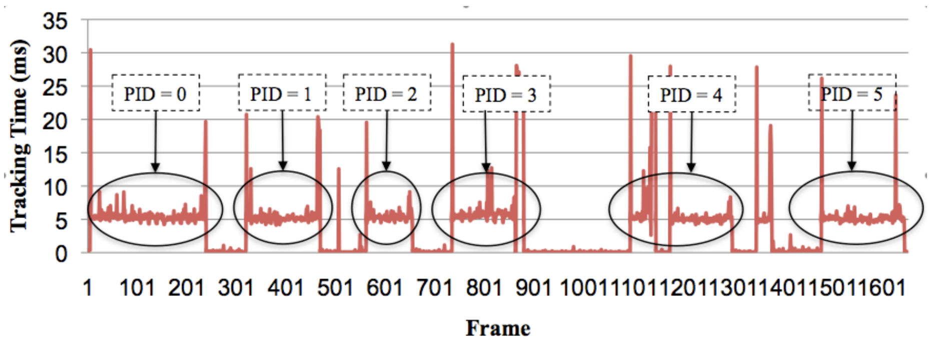
In this section, we explain practical implementation issues of the proposed algorithm and show experimental results. We used a 2.66 GHz core 2 duo CPU with a GTX280 NVIDIA graphic card for our experiments. The camera resolution was 640 × 480 pixels and it supported up to 60 frames per second for rendering image onto a screen. We used an ordinary book consisting of figures and texts. No special markers were attached onto pages in the book. We implemented an AR book system using OpenSceneGraph [18]. The OpenSceneGraph allows to manage virtual contents efficiently. And also the proposed tracking algorithms were implemented using OpenCV [19] library. We exploited Graphics Processing Unit (GPU) implementation of SIFT [20] for increasing the speed of the page recognition.

4.1 Page Recognition and Marker-less Tracker Performance

We measured the processing time in the page recognition. In total 30 pages were registered in advance and the features mentioned in 3.1 were extracted. The time for the registration was only 3 seconds for 30 pages with the help of GPU powers. This is the significant contribution of the proposed tracking algorithm compared to the other tracking algorithm which required about 1 minutes for one page training [4]. Figure 4 shows the selected results when a user turns a page step by step. We augmented different virtual contents with respect to each page ID. The page is successfully detected in all cases. As shown in Figure 4(b), the overall frame rate was more than 30 fps.



(a)

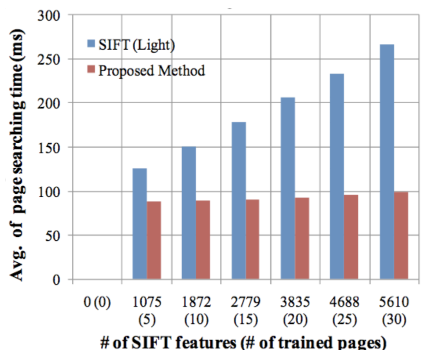


(b)

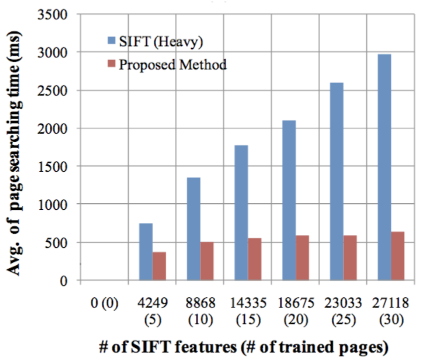
Fig. 4. Results of the page recognition: (a) page-dependent contents and page ID were augmented on each page (b) corresponding performance results for the sequence

The performance results of the page recognition in background thread are shown in Figure 5. We compared the proposed algorithm with the sequential matching test, which showed the worst recognition results. The number of SIFT features depends on the octave and scale parameters. We carried out the comparison with two conditions; light and heavy. As shown in Figure 5, the proposed method does not drastically increase the page searching time according to the number of SIFT feature points from the registered pages.

To show the performance of the proposed tracker, we measured the time and the accuracy for the proposed tracker. As shown in Figure 6(a), the tracker is robust to distance changes and rotation of a camera. Figure 6(b) shows the average time spent in each step of the proposed algorithm. The proposed tracker consists of three big steps; preparation, frame-to-frame matching, and pose computation. The preparation step includes capturing camera image, checking the current page number obtained from the background thread, and copying points. Note that the coarse-to-fine matching time depends on the number of tracked



(a)



(b)

Fig. 5. Performance results of the page recognition compared to SIFT sequential matching: (a) SIFT light condition: octave(2) and scale(2) (b) SIFT heavy condition: octave (3) and scale(3)

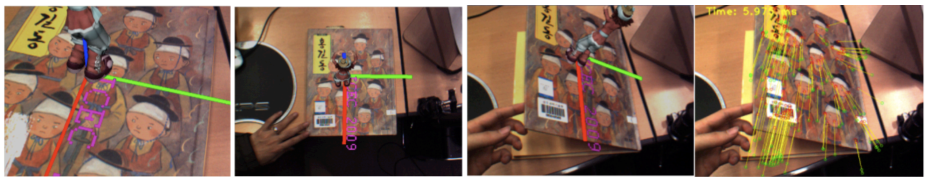
points and the size of the searching window. We used the maximum 200 points per page and 8×8 window size for this experiment. We observed that our algorithm runs at between 4ms and 8ms in most cases. It is as shown in Figure 6(c).

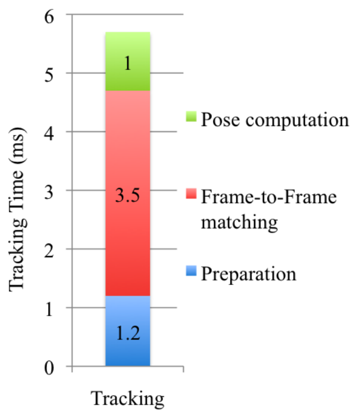
4.2 Application to Multi-layer Authoring

We built a multi-layer authoring system using the proposed tracker, as explained in Section 3.4. The environmental setup and two examples are shown in Figure 7. We generated two layers and mapped a figure and a video file to each layer. In the authoring window, users could manipulate the defined layers and see the mapped files in a text format. In the AR viewer window, as a result, users could see the AR contents which had been configured before. The benefit of the multi-layer authoring with the proposed marker-less tracking compared to marker-based authoring is that we can use the context of figures and texts on the page when a user make his/her own story.

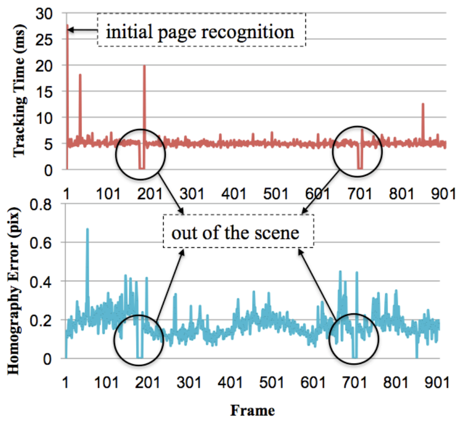
5 Conclusions and Future Works

We proposed the speed-improved marker-less tracker and the multi-layout authoring method. We showed that the proposed multi-layout authoring was useful to facilitate the content-aware augmentation. Especially, the proposed tracker outperformed the existing methods in time aspect. By using the proposed tracker and authoring tool, we could have a rapid prototype of an AR book application. In the future, we will consider edge features for more robust tracking results. The tracker with a book of more than 30 pages will be tested and analyzed. And also we will consider the interaction authoring with a multi-layer interface. The


(a)



(b)



(c)

Fig. 6. Results of the marker-less tracker performance: (a) axis augmentation and tracked points (b) average time spent in the camera tracking thread (c) corresponding time and accuracy results

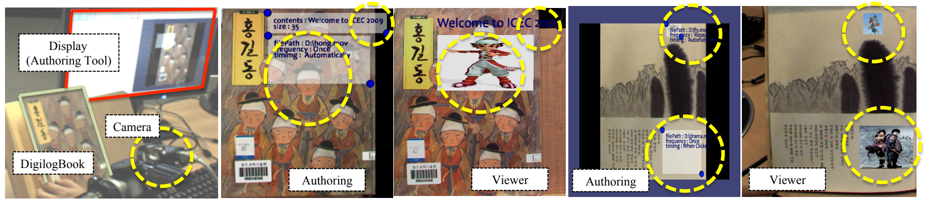


Fig. 7. Snapshots of the multi-layer authoring tool: (from the left) environmental setup, mapping a figure and a video to two layers, example of another page authoring

proposed methods will be used not only in AR book applications, but also in other AR applications which require multiple planes tracking.

References

1. Billinghurst, M., Kato, H., Poupyrev, I.: The magicbook - moving seamlessly between reality and virtuality. Computer Graphics and Applications, IEEE **21**(3) (May 2001) 6 – 8

2. 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 (Jan 2007) 475 – 484
3. Scherrer, C., Pilet, J., Fua, P., Lepetit, V.: The haunted book. Mixed and Augmented Reality, 2008. ISMAR 2008. IEEE/ACM International Symposium on (2008) 163 – 164
4. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on **2** (May 2005) 775 – 781 vol. 2
5. Yang, H.S., Cho, K., Soh, J., Jung, J., Lee, J.: Hybrid visual tracking for augmented books. Entertainment Computing - ICEC 2008 **5309** (Jan 2008) 161–166
6. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) (Nov 2004) 91–110
7. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on **2** (Sep 2005) 1508 – 1515 Vol. 2
8. Vacchetti, L., Lepetit, V., Fua, P.: Stable real-time 3d tracking using online and offline information. Pattern Analysis and Machine Intelligence, IEEE Transactions on **26**(10) (Oct 2004) 1385 – 1391
9. Reitmayr, G., Drummond, T.: Going out: robust model-based tracking for outdoor augmented reality. Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on (Oct 2006) 109 – 118
10. Ozuysal, M., Fua, P., Lepetit, V.: Fast keypoint recognition in ten lines of code. Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on (May 2007) 1 – 8
11. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: Real-time single camera slam. Pattern Analysis and Machine Intelligence, IEEE Transactions on **29**(6) (Jun 2007) 1052 – 1067
12. Williams, B., Klein, G., Reid, I.: Real-time slam relocalisation. Computer Vision, 2007. ICCV 2007. IEEE International Conference on (Sep 2007) 1 – 8
13. Castle, R., Gawley, D., Klein, G., Murray, D.: Video-rate recognition and localization for wearable cameras. British Machine Vision Conf (Jan 2007)
14. Klein, G., Murray, D.: Parallel tracking and mapping for small ar workspaces. Mixed and Augmented Reality, 2007. ISMAR 2007. IEEE/ACM International Symposium on (Oct 2007) 225 – 234
15. Bay, H., Ess, A., Tuytelaars, T., Vangool, L.: Speeded-up robust features (surf). Computer Vision and Image Understanding **110**(3) (Jun 2008) 346–359
16. Lee, T., Hollerer, T.: Hybrid feature tracking and user interaction for markerless augmented reality. Virtual Reality, 2008. VR 2008. IEEE (Feb 2008) 145 – 152
17. Pilet, J., Geiger, A., Lagger, P., Lepetit, V., Fua, P.: An all-in-one solution to geometric and photometric calibration. Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on (Sep 2006) 69 – 78
18. : Openscenegraph, http://www.openscenegraph.org
19. : Open computer vision library, http://sourceforge.net/projects/opencvlibrary/
20. : Siftgpu, http://cs.unc.edu/ ccwu/siftgpu/