Efficient Inter-camera Management for Multiple Objects Tracking in Mobile AR Environments *†

Woonhyuk Baek‡ Woontack Woo§

GIST U-VR Lab.
Gwangju 500-712, S. Korea

ABSTRACT

This paper presents a real-time multiple objects tracking based on inter-camera approach for mobile devices. The key idea of our approach is to accommodate the pose of each cameras by sharing the results of tracking on heterogeneous inter-camera. Our approach accomplishes the time-consuming multiple objects tracking on a server while the mobile device tracks only a single object and then shares the result of the tracking with the mobile device. The proposed approach enables mobile devices to track multiple and complex objects with limited network bandwidth. As experimental results, the proposed inter-camera approach has shown that it is possible to track occluded objects. The proposed approach can be effectively used for mobile augmented reality applications that need to consider multiple and complex objects like desktop AR applications; then it is easy to extend the desktop approach to mobile AR applications.

Index Terms: H.5.1 [Information Interface and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.4.1 [Image Processing and Computer Vision]: Scene Analysis—Object recognition, and tracking

1 INTRODUCTION

As mobile phone use has spread throughout the world and as the phones are increasingly equipped with cameras and wireless capability, augmented reality (AR) has also become increasingly attractive. Mobile AR applications have generally used sensor-based tracking, but this method lacks the ability to combine the real and the virtual world in 3D. Generally, the purpose of object tracking in AR is to compute a relative camera pose, represented in a rotation and a translation to 3D. Object tracking using natural features has a high accuracy in 3D and it has now been well explored by AR applications [15]. However, mobile phones have limited computational performance and a small memory size. It is not enough simultaneously to track multiple objects and to render the contents in 3D.

Mobile AR has applied the server-client approach to solving the limitations of mobile phones. The AR-PDA [5] and other mobile AR projects[12] have used a server-client approach. They send the video stream by mobile radio communication to the AR-server. The server recognizes the object by analyzing the image, which is added to the video stream (like 3D content) and then sent back to the client. However, the mobile radio communication network bandwidth is not enough to transfer the client camera image to the server in real-time. Moreover, the lack of network bandwidth causes a delay in tracking. In particular, even if there is enough network bandwidth, the quality of mobile phone cameras is not good enough to recognize multiple objects.

In this paper, we separate the server and mobile phone devices. The server tracks multiple objects while the mobile phones tracks only a single object. The server detects and tracks multiple objects using cameras, and shares only the pose of the objects with mobile devices. Then, the mobile devices restore the pose of objects by tracking only one of them. Our approach is able to efficiently deal with several objects using computational power of the server and saving network bandwidth. The proposed inter-camera approach supports real-time multiple objects tracking, including hard track objects, such as an occluded object.

In the remainder of the paper, we discuss related work on mobile tracking for mobile AR applications in Chapter 2. Chapters 3 and 4 describe our method and present our experimental results. We conclude this paper with suggestions for future work in Chapter 5.

2 RELATED WORKS

Mobile AR applications are generally used in Global Positioning Systems (GPS) and electronic compasses, and inertial sensors are widely used in outdoor tracking and AR systems [13, 7]. Compared to model-based vision tracking [8] and feature-based vision tracking[18], these tracking technologies are often more robust and accurate.

Indoor mobile AR systems often use fiducial markers [20] rather than sensor data to simplify feature detection and matching. However, many markers become difficult to recognize if they are offscreen or occluded. Therefore, natural feature tracking algorithms are better than marker-based approaches. They use interest point detectors and matching schemes to associate 2D locations in the images with pre-defined 3D locations in the reference model. SLAM-based methods have been used in unknown environments to realize mobile AR [6].

Mobile AR applications require multiple objects tracking, using natural features, in order to realize indoor mobile AR systems. The natural feature-based multiple objects tracking algorithm[11] is closed to our multiple objects tracking approach, which detects multiple targets over multiple frames, but without considering mobile devices. D. Wagner's algorithm [19] used multiple objects tracking by means of natural features on a mobile phone. It was able to track multiple objects on a mobile phone, but it was not able to realize mobile AR applications due to the heavy computation costs of calculating multiple objects tracking on only a mobile device. The AR-PDA project [5] used the computational power of a server to solve the limitations of the mobile device. This project sends image streaming to a server. But it is unreasonable to expect a real-time response that is, the results of multiple objects tracking because of lack of network bandwidth.

Our method overcomes the limitations in the performance of mobile devices by using an inter-camera approach. The proposed

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

†This research was supported by the Global Frontier project of MEST in S.Korea.

‡e-mail:wbaek@gist.ac.kr

§e-mail:wwoo@gist.ac.kr

20th International Conference on Artificial Reality and Telexistence (ICAT2010), 1-3 December 2010, Adelaide, Australia
ISBN: 978-4-904490-03-7 C3450

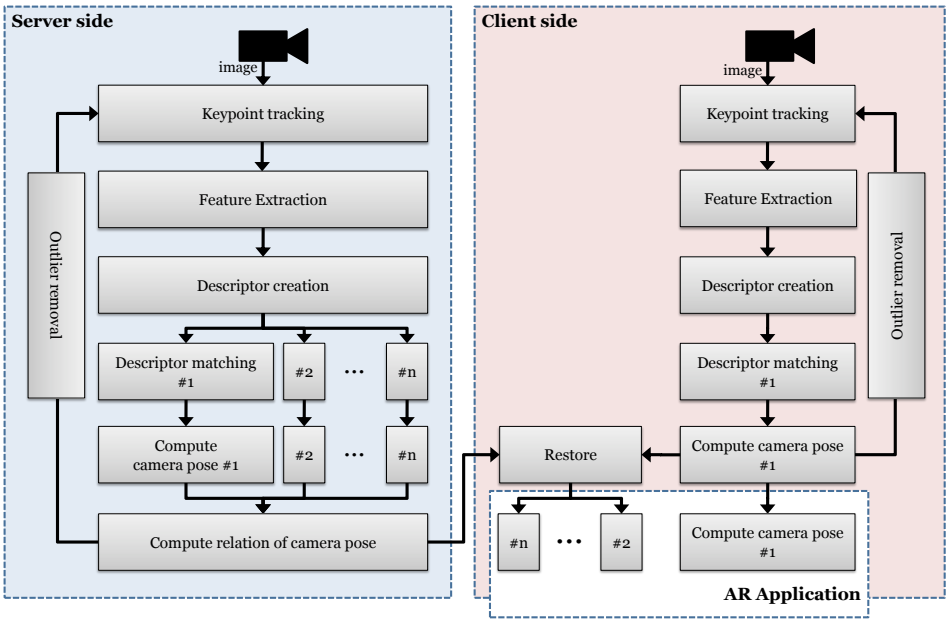


Figure 1: Overall flowchart of the proposed inter-camera approach

method differs from [5][12] in which we added cameras to a server to reduce information transfer.

3 Inter-camera Approach

Multiple objects tracking using natural features in mobile devices, such as smart phones, is an essential part of realizing mobile AR applications. But the mobile devices have limited performance compared to desktop PCs, low throughput, limited storage, and slow memory. Mobile devices lack the processing performance to track multiple objects. Specifically, an arbitrary shape object tracking algorithm needs a high-performance processor, such as a multi-core CPU and GPU. Therefore, we add a server to make up for the lack of processing performance in mobile devices.

As shown in Fig. 1, the proposed inter-camera approach divides server and client. The server tracking is focused on multiple objects; it tracks using natural features and calculates relations between multiple objects using the results of tracking as, camera pose. Client tracking is single object tracking only, using natural features and restoring the camera pose of the other objects, using received data that is relational data from the server.

3.1 Object detection

Multiple objects tracking using natural features is a must for realizing server tracking. It has to guarantee processing speed in real-time. This part is important for server tracking and client tracking. But client object detection considers a single object that reduces part of multiple objects.

The natural feature tracking is composed of four steps (see Fig. 2): feature extraction, feature description, feature matching, and pose estimation. The feature points are enough interesting points of an image captured by a camera. It is robust enough to recognize a point that is the same point in different images. First, we do not estimate the feature points scale using the FAST corner detector [14]. To reduce the time consumption for scale estimation, we acquire the database containing descriptors from the multi-scale

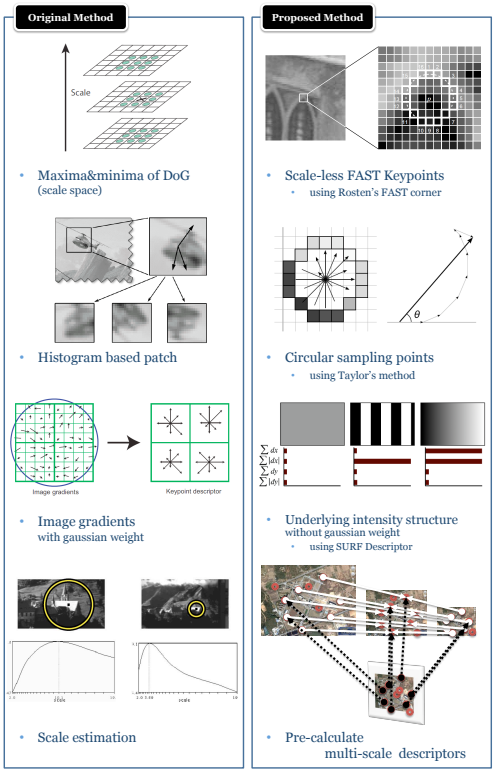


Figure 2: Comparison of object detection using natural features between the original method(SIFT[9], Wagner's work[18], Scale-estimation[17]), and the proposed method(FAST[14], corner and SURF[1]).

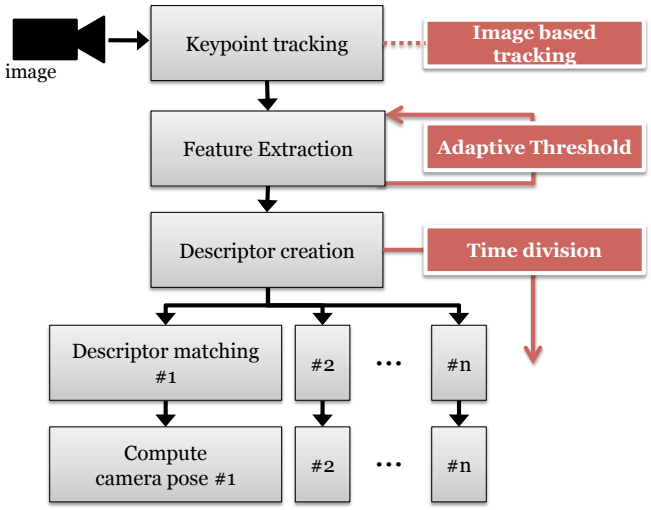


Figure 3: Multiple objects using natural features with guaranteed process time

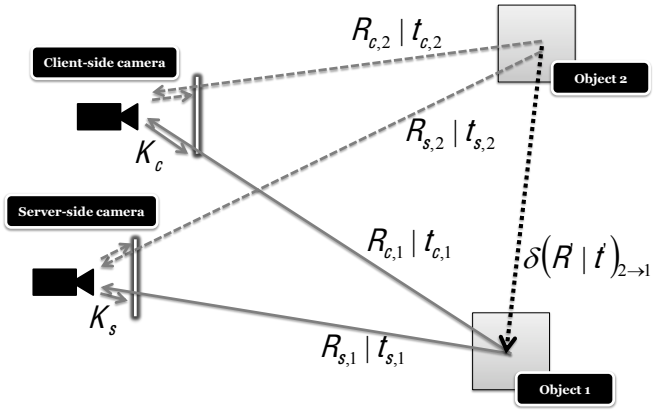


Figure 4: Relation between objects independent server and client cameras

feature points in advance. Second, we estimate the orientation of each feature point based on the image patch that is the area around the feature point [16]. Then, by exploiting the calculated orientation, it makes each feature invariant to the rotation. The patch is rotated to compensate for the rotation. Based on the rotated patch, a descriptor is created. That is the SURF[1]-like descriptor.

The descriptors for all feature points in the input image have been created. They are matched with the descriptors in the database. The database is pre-computed for every feature point and descriptor from the registered image. The descriptor database constructs a tree structure, using KDtree [2], to find nearest descriptor quickly. Finally, we estimate a camera pose from the correspondence points between feature points in the input image and feature points in the database. Moreover, to support the robust and fast feature tracking, we use image-to-image feature tracking method to reduce the processing time for physical object tracking.

3.2 Multiple objects tracking

Our proposed detection algorithm, using the natural features in section 3.1, is enough to track several objects. However, we should guarantee process time in real-time for multiple objects tracking. So, we add several techniques to increase the speed, less effect on the number of tracked objects (see Fig. 3).

We apply an adaptive threshold to keep the number of features at the feature extraction step for every frame and the time division approach to guarantee the processing time because the process time for the feature extraction step and the description step are dependent on the number of features. The feature matching step is also highly dependent on the number of objects. We apply the time division approach, but this approach has a problem. The problem is the term of divided times. To connect this term, we keep the feature points, using the image-to-image tracking method, which is independent of the number of objects since the image-to-image tracking method depends on the image resolution. Therefore, this method is less influenced by the entire processing time for object tracking.

3.3 Pose restoration

The server tracks multiple objects and calculates the relations of the camera pose to restore the camera pose for each object in the

client. But the server and client cameras may have different lenses and hence different intrinsic parameters.

$$K_s \neq K_c. \tag{1}$$

Therefore, we divide the camera pose P into intrinsic matrix K and extrinsic matrix $(R|t)$ to remove the effect of the intrinsic parameters on each side

$$P = K(R|t). \tag{2}$$

Fig. 4 shows the camera pose of multiple objects $(R_{s,1\ldots n}|t_{s,1\ldots n})$ in the server. We can calculate the relation between objects $\delta\left(R^{'}|t^{'}\right)$ that is independent not only of the intrinsic parameters but also of the pose of the camera in the server using Eq. 3;

$$\delta\begin{pmatrix} R^{'} & t^{'} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{s,1} & t_{s,1} \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} R_{s,2} & t_{s,2} \\ 0 & 1 \end{pmatrix} \tag{3}$$

The relation between objects $\delta(R^{'}|t^{'})$ is independent of the server camera, not only of the pose, but also of the intrinsic parameters. So, the client can restore the camera pose of *object 2* using the extrinsic matrix of *object 1*, and the relation between the objects may be expressed as

$$\begin{pmatrix} R_{c,2} & t_{c,2} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{c,1} & t_{c,1} \\ 0 & 1 \end{pmatrix} \times \delta \begin{pmatrix} R^{'} & t^{'} \\ 0 & 1 \end{pmatrix} \tag{4}$$

Then, it is possible to restore the camera pose of *object 2* using Eq. 4, as shown in Eq. 5:

$$P_{c,2} = K_c(R_{c,2}|t_{c,2}). \tag{5}$$

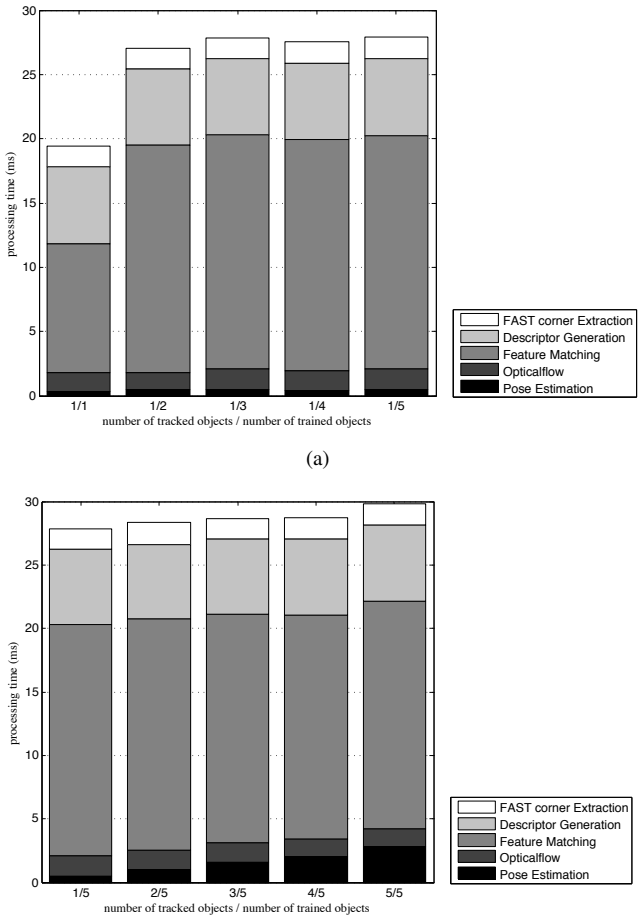


Figure 8: Results of the multiple objects tracking performance ((a) performance difference of the number of trained objects; (b) performance difference of the number of tracking objects)

4 Implementation and Results

In this section, we explain the practical implementation issues of the proposed approach and show experimental results. We used a general PC with a 2.93 GHz CPU for a server and a TOSHIBA TG01 equipped with Windows Mobile 6.5 OS and snapdragon as the ARM-based 1 GHz CPU for the client. The camera in server resolution was 640x480 pixels, and it supported up to 30 frames per second for tracking. The camera in the smart phone was 320x240 pixels and produced up to 15 frames per second for tracking and rendering the image onto a screen. We implemented an inter-camera system using OpenCV[4] library and oscpack[3] as the UDP networking library. We implemented different code paths for the PCs and mobile phones based on the flowchart (See Fig. 1). In the server, we concentrate on the accuracy of the camera pose results. In the client that is a smart phone, we focus on the speed of the natural feature tracking.

4.1 Server tracking

The server tracking system (see Fig. 5) that supports multiple target tracking needs to run both the multi-target detection as well as the tracking task that each targets simultaneously: the tracker must estimate the poses of all previously detected targets while another tracker detects other targets.

The server tracking system will guarantee processing time less effect on the number of tracking objects. Fig. 8 shows the performance of the multiple objects tracking. The results of the performance are less effect on the number of tracking objects and trained objects.

Fig. 8(a) shows that the change in processing time is dependent on the number of trained objects. If the number of trained objects is larger than 2, this graph shows that the processing time is less effect on the number of trained objects, and a point of difference appears on the number of trained objects between 1 and 2 because multiple objects tracking is needed to switch the search tree and manage the feature points.

In case of a change in the number of tracked objects, Fig. 8(b) has increased the processing time a little by changing the number of tracked objects because the server tracking is detecting multiple objects over frames. That, we can keep the entire processing time in real-time.

4.2 Mobile tracking

The mobile tracking system (see Fig. 6) that supports sufficiently speedy tracking of the single target in real-time needs to run both the detection and the tracking tasks simultaneously while the tracker estimates the pose of all previously detected targets in a few frames. Code optimization is one of the important parts of a mobile tracking system [10]. This work aims at accelerating single object tracking using source code-level optimization based on the multiple objects tracking method.

We measured the processing time to recognize a planar object

Figure 5: Results of the multiple objects tracking in server

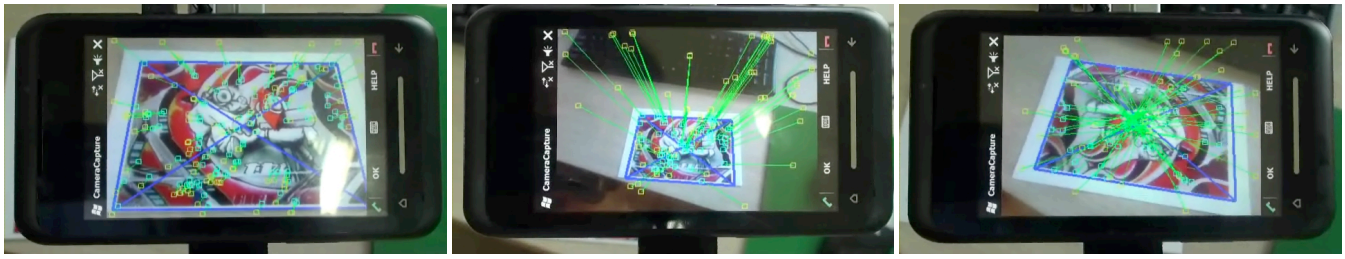
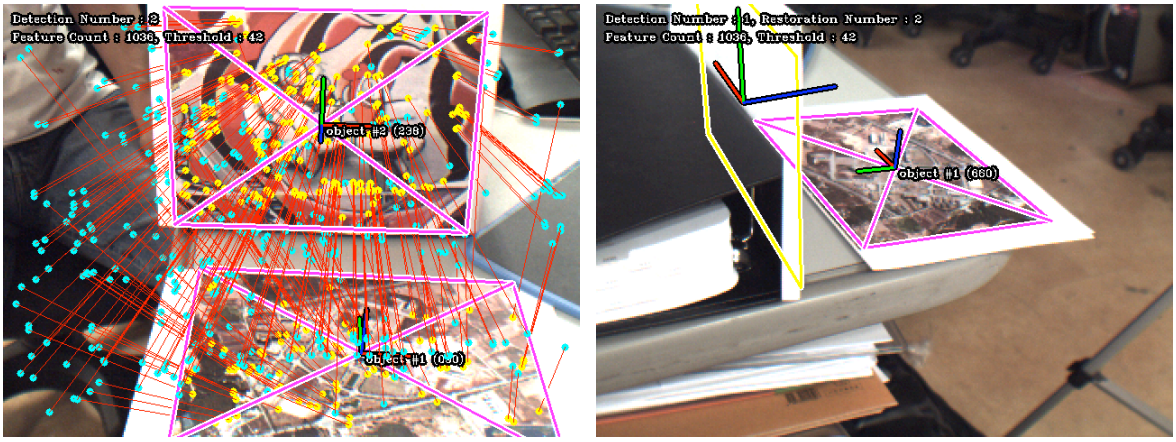


Figure 6: Results of the object tracking in mobile phone



(a) (b)

Figure 7: Result of calculation of relation and restoration ((a): multiple objects tracking in server, (b): restoration in client)

Table 1: Performance results of the single object tracking on mobile phone (ms)

Algorithms	Feature Tracking	Feature Extraction Describing	Feature Matching	Pose Estimation	Sum
Process Time	35.73	10.24	36.71	15.83	98.51

Table 2: Average difference between result of restoration from server and result of tracking in client (units : mm)

	x-axis	y-axis	z-axis
Average of error	3.632	12.553	18.159
Standard deviation of error	2.795	4.030	5.506

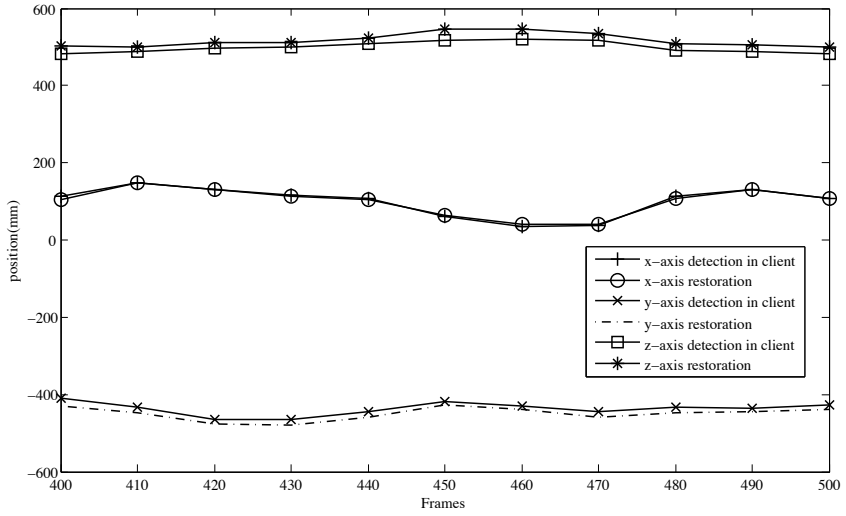


Figure 9: Difference between result of restoration from server and result of tracking in client each axis in sampled range

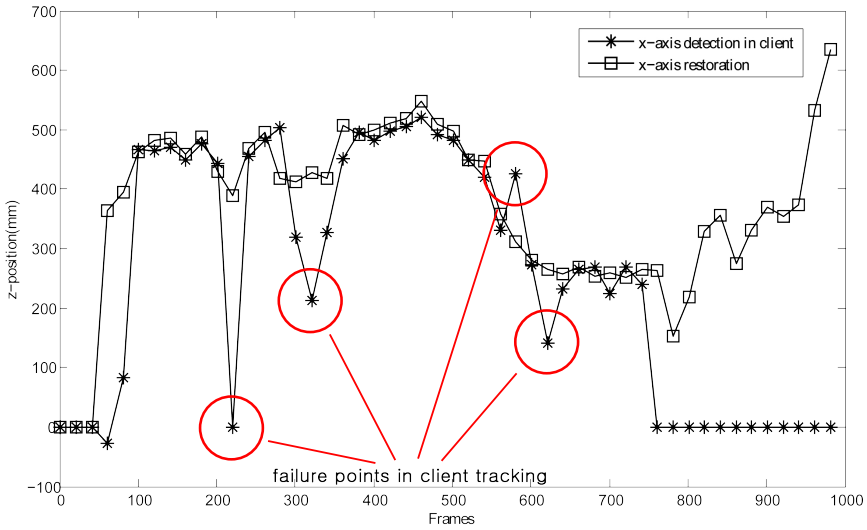


Figure 10: Difference between result of restoration from server and result of tracking in client z-axis

and to track the natural features (see Table 1). The average processing time per frame is around 100ms on a smart phone, resulting in a potential frame rate of 10 frames per second. So we can possibly track the planar target in real-time to restore the poses of multiple objects and to realize an AR application.

4.3 Relation & restoration

We mentioned an inter-camera approach in using the relation between objects. Fig. 7(a) shows the server tracking that represents the multiple objects tracking and calculation of the relationships. Fig. 7(b) shows client tracking that consists of restoration using the relations from the server. In particular, the yellow rectangle and inside 3D axis represent the pose of a hard-to-track object.

We measured the accuracy of the restoration using relation data from the server in sampling data that is well detected by both the server and client tracking algorithm (See Fig. 9 and Table 2). Each group of lines is a restoration of the camera position from the server and the detected camera position on the client side. The result of difference camera poses between server and client is quite accurate (within 2 centimeters) and is, moreover, stable. Fig. 10 at point of circle and after 760 frames shows points of cannot tracked the object in the client, but the server tracker is able to track that object. Thus, the client can restore the pose of that object from the relation data obtained from the server.

5 Conclusions

We have proposed an inter-camera approach that supports real-time multiple objects recognition and tracking in mobile devices. We showed that the proposed inter-camera tracking is useful for tracking in mobile devices. By using the proposed inter-camera approach, we have made it possible to realize mobile AR application. In the future, we will consider multiple cameras in the server and multi-user' cameras to support more robust tracking. Especially, in the case of multi-users, it is possible to use the results of object tracking in another client's mobile devices, and it is also possible to realize interactive mobile AR applications.

REFERENCES

[1] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV 2006*, 2006.

[2] J. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in highdimensional spaces. In *CVPR 1997*, 1997.

[3] R. Bencina. oscpack. http://www.audiomulch.com/ rossb/code/oscpack/.

[4] G. Bradski. The opencv library. http://opencv.willowgarage.com/, 2000.

[5] J. Gausemeier, J. Fruend, C. Matysczok, B. Bruederlin, and D. Beier. Development of a real time image based object recognition method for mobile ar-devices. In *Inetrnational Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction*, pages 133–139, 2003.

[6] K. Georg and M. David. Parallel tracking and mapping on a camera phone. In *ISMAR 2009*, 2009.

[7] B. Jiang, U. Neumann, and S. You. A robust hybrid tracking system for outdoor augmented reality. In *VR 2004*, pages 3–10, 2004.

[8] G. Klein and T. W. Drummond. Robust visual tracking for noninstrumented augmented reality. In *ISMAR 2003*, pages 113–122, 2003.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110, 2004.

[10] N. Mcallister. Rethinking code optimization for mobile and multicore. http://www.infoworld.com/d/developer-world/rethinking-code-optimization-mobile-and-multicore-505, 2009.

[11] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In *ISMAR 2008*, 2008.

[12] W. Pasman and C. Woodward. Implementation of an augmented reality system on a pda. In *in Proc. The Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR03*, pages 276–277, 2003.

[13] W. Piekarski, B. Gunther, and B. Thomas. Integrating virtual and augmented realities in a outdoor application. In *IWAR 99*, pages 45–54, 1999.

[14] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *ECCV2006*, pages 430–443, 2006.

[15] J. P. Suya, J. Park, S. You, and U. Neumann. Natural feature tracking for extendible robust augmented realities. In *International Workshop on Augmented Reality (IWAR)'98*, 1998.

[16] S. Taylor, E. Rosten, and T. W. Drummond. Robust feature matching in 2.3 microseconds. In *IEEE Workshop on Feature Detectors and Descriptors*, 2009.

[17] T. Tuytelaars and K. Mikolajczyk. K.: Local invariant feature detectors: A survey. pages 177–280, 2008.

[18] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR 2008*, pages 125–134, 2008.

[19] D. Wagner and B. Schmalstieg, D.and Horst. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *ISMAR 2009*, 2009.

[20] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *Computer Vision Winter Workshop*, pages 139–146, 2007.