Cloud Computing Based Mobile Augmented Reality Interactive System

Pei-Hsuan Chiu, Po-Hsuan Tseng*, and Kai-Ten Feng
Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan
*Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan
mean11809.cm01g@nctu.edu.tw, *phtseng@ntut.edu.tw, and ktfeng@mail.nctu.edu.tw

***Abstract*—Augmented reality (AR) technology is mainly composed of feature extraction, feature points matching and 3-D object drawing to impose virtual information onto the real camera. Hand tracking method is designed to realize interactive AR system to further enhance user experience (UX). However, feature extraction, feature points matching, and hand tracking all require high computational cost. To achieve real-time implementation on mobile, we proposed a Mobile Augmented ReAlity Interactive System based on cloud computing (MARAIS). Cloud side is responsible for heavy tasks that need sufficient computational capabilities, and device side performs sensing, detecting and displaying. Device/cloud architecture has been designed to maintain tolerable processing delay with minimum communication overhead. To validate the suitability for real-time interaction, MARAIS is combined with a picture book to realize a digital learning system. Implementation of interactive AR system based on cloud computing eases the requirement of data storage, memory, and computational power at device side.**

***Index Terms*—interactive augmented reality (AR) system, cloud computing, particle filter.**

I. Introduction

Augmented reality (AR) [1] is a technology that enhances the perception by combining the virtual information with the sensing environments. AR plays a role between VR and actual reality, which combines virtual information such as images, videos, and 3-D object with the real camera view of mobile platforms. Thus, AR becomes a feasible technology for enhancing user experiences (UX).

To realize an AR application, pattern recognition that includes feature detection, feature description, and feature points matching are required. Features represent a set of points carrying unique information in an image, such as corners. Feature detection decides if each pixel in an image is a feature point or not, by using image information around it with computer vision and image processing techniques. There are many approaches to detect feature points such as FAST corner detection [2], SIFT feature detection [3], SURF feature detection [4] and so on. Feature description creates a descriptor to illustrate each feature point found at the step of feature detection. Common descriptor extractor such as SIFT, SURF, and BRIEF [5] extractor can be utilized to calculate feature description. After obtaining descriptors of feature points, feature descriptors of captured image are compared to those of target image in the database by feature points matching.

SIFT or SURF detector and descriptor have shown to be successful in object recognition. However, both methods require a lot of computational power and are not suitable for real-time applications. ORB [6] is the combination of FAST feature detector and modified steered BRIEF feature descriptor extractor. Compared to other schemes, ORB uses less number of detected feature points and shorter detection time to maintain better performance.

Interacting with 3-D AR objects is designed to bring in more UX. To perform real-time hand tracking for interaction recognition, hand tracking with the color gloves [7] is easy for implementation and with high precision. However, it is not convenient for user in many situations to wear cloth gloves. Tracking object using background subtraction [8] can identify the position of moving object, but it requires a steady background which is unsuitable for background captured by mobile camera. Depth image [9] is a good information to identify foreground and background, but it needs a time-of-flight camera or a stereo camera. Color-based hand tracking can be implemented on many mobile platforms because it only requires a color camera. However, if there are some skin-color objects in the background, the system will lose track of the target due to background noises.

With the rapid development of mobile Internet, cloud computing can resolve the problem of computational complexity. For example, a book finding system proposed in [10] utilizes a mobile camera to capture images of a book spine by taking a snapshot and recognizes the features at server side. Information such as book title, author information, and publisher information are sent back to mobile device. However, in the AR interactive system, the features are sent continuously to server for real-time interaction, which is a more challenging task in system design.

Qualcomm's Vuforia [11] is one the most representative software platform for AR development on mobile devices. However, only upper layer application programming interfaces (APIs) are provided to developers, which cannot fulfill our requirement to integrate AR with hand tracking scheme. Besides, the image database can only be created from their website which limits the design flexibility for cloud services.

In this paper, we utilize OpenCV [12] and OpenGL [13]

[1]This work was in part funded by the Aiming for the Top University and Elite Research Center Development Plan, NSC 102-2221-E-009-018-MY3, NSC 101-2219-E-009-027, NSC 102-2221-E-027-004, the MediaTek research center at National Chiao Tung University, and the Telecommunication Laboratories at Chunghwa Telecom Co. Ltd, Taiwan.

978-1-4799-3083-8/14/$31.00 ©2014IEEE

to realize an AR interactive system on Android-based mobile platforms. When a user adopts a mobile camera to capture a scene containing the target image, the proposed system will generate a 3-D object from the camera view on the display over the target image. At the same time, the user can interact with the 3-D object. However, performing both real-time hand tracking and pattern recognition processes for AR require sufficient computational power. Besides, a large size of storage is required for the database of pattern recognition. To ease the computation and storage requirement of mobile platforms, we introduce cloud concepts to an AR interactive system for the proposed Mobile Augmented ReAlity Interactive System based on cloud computing (MARAIS).

This paper consists of five sections. In section II, the system architecture of MARAIS is described. Section III presents the core components of MARAIS system in details. Section IV shows the implementation results and discussion with examples from picture books that are followed by the conclusion in section V.

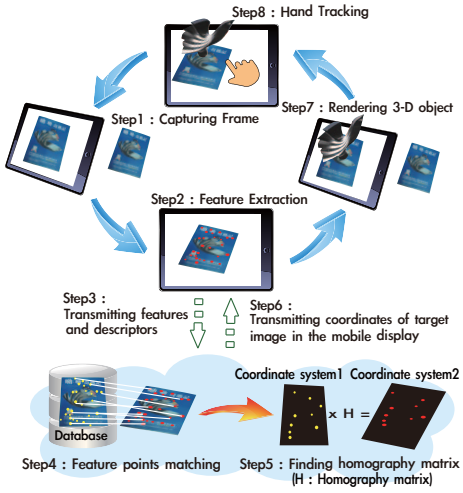


Fig. 1. System architecture of MARAIS.

II. System Architecture

The framework of MARAIS is depicted in Fig. 1. To decrease the loading of mobile devices, the proposed MARAIS is constructed with device-cloud model. The procedure that requires more computing resources, i.e., homography matrix finding, is implemented at the cloud side, and the sensing and detecting part are processed at the device side. In Fig. 1, the procedure of MARAIS is summarized step-by-step, which is listed as follows.

1) At the device side, user captures the real view with mobile camera.
2) Feature detection and feature description procedure extract the features from the captured images.
3) The mobile transmits these feature points and descriptors, which have smaller data size compared to the captured frame, to cloud through the Internet.

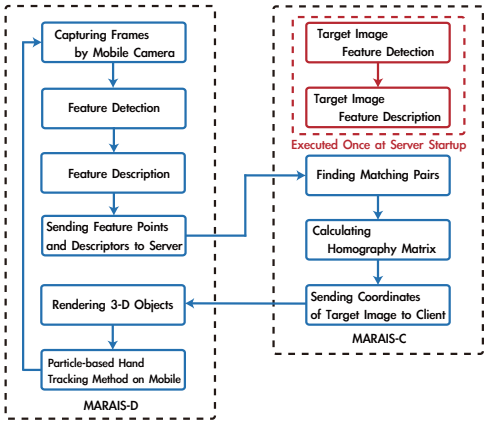


Fig. 2. Flow chart of MARAIS.

4) After receiving features from the device side, feature points matching compares these features from captured image with features from database to find matching pairs.
5) The system finds a homography matrix which can transform coordinates between two coordinate systems according to these matching pairs.
6) The matching result that contains the location information of target image in camera view is sent back to device.
7) The device renders 3-D object through the information from cloud by OpenGL.
8) The captured frame will be preprocessed by OpenCV and hand tracking method estimates the location of user's hand at device side.

Following these steps, the interaction between user and 3-D object is shown on mobile camera view.

III. Mobile Augmented ReAlity Interactive System based on Cloud Computing (MARAIS)

As shown in Fig. 2, the proposed system is divided into two parts, which are MARAIS at device side (MARAIS-D) and MARAIS at cloud side (MARAIS-C). The details of their components are described as follows.

A. MARAIS at Device Side (MARAIS-D)

In Fig. 2, the first step at device side is capturing color frames with mobile camera. The second and third steps are feature detection and description where the ORB algorithm is utilized to find feature points, based on oriented FAST and rotated BRIEF. MARAIS-D sends these feature points and descriptors to MARAIS-C. After receiving information back from MARAIS-C, MARAIS-D renders the corresponding 3-D object onto the camera view. The last step is Particle-based HANd Tracking methOd on Mobile (PHANTOM) which provides user interaction with the 3-D object. In this section, three major parts including feature detection and description, and PHANTOM are described.

1) Feature Detection and Description: In feature detection, grayscale intensity difference between center pixel and pixels in a circular ring is compared with a predefined threshold to detect if this center pixel is a FAST feature point. Meanwhile,

the orientation of the detected feature points can be calculated by intensity centroid [14]. The moment of an image patch is defined as

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \ , \tag{1}$$

where x and y are coordinates of pixels in a patch and $I(x,y)$ is the 8-bit grayscale intensity at (x,y). The intensity centroid is calculated by

$$C(x,y) = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) = (c_x, c_y) \ . \tag{2}$$

Providing that the pixel (x_f, y_f) is a detected feature point, the orientation of this patch is recorded as

$$\theta(x_f, y_f) = tan^{-1}(\frac{c_y}{c_x}) \ . \tag{3}$$

When feature points are detected, a descriptor is required to describe each feature point. Based on a set of binary tests, a bit string of a patch associated with a detected feature point is utilized for the conventional BRIEF descriptor. A binary test τ of a patch $\mathbf{p}$ associated with a detected feature point is defined by:

$$\tau(\mathbf{p}; \mathbf{k}, \mathbf{k}') := \begin{cases} 1 : I(\mathbf{k}) < I(\mathbf{k}') \\ 0 : I(\mathbf{k}) > I(\mathbf{k}') \end{cases} \ . \tag{4}$$

$\mathbf{k}$ and $\mathbf{k}'$ represent the test locations in the patch $\mathbf{p}$, where the test locations in the patch $\mathbf{p}$ is defined by BRIEF descriptor. Note that $\mathbf{k}$ is represented by 2-D coordinates of the pixel. Therefore, $I(\mathbf{k})$ is the intensity at pixel $\mathbf{k}$.

Steered BRIEF further combines traditional BRIEF and the orientation $\theta(x_f, y_f)$ of each feature point. With the corresponding rotation matrix

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta(x_f, y_f) & -\sin\theta(x_f, y_f) \\ \sin\theta(x_f, y_f) & \cos\theta(x_f, y_f) \end{bmatrix} \ , \tag{5}$$

the steered version of the test location $\mathbf{k}_{\theta_i}$ can be obtained as

$$\mathbf{k}_{\theta_i} = \mathbf{R}_\theta \mathbf{k}_i. \tag{6}$$

By rotating the test locations, the feature is defined as a vector of n binary tests by substituting (6) to (4):

$$f_n(\mathbf{p}) := \sum_{1 \leqslant i \leqslant n} 2^{i-1} \tau(\mathbf{p}; \mathbf{k}_{\theta_i}, \mathbf{k}'_{\theta_i}). \tag{7}$$

In [6], a learning method is developed for choosing better subset of binary tests, which is based on a greedy search for a set of uncorrelated tests. The feature $f_n(\mathbf{p})$ well describes the detected feature point in binary format and this descriptor is robust for the image orientation.

2) PHANTOM: Since hand tracking process is required in every camera frame at each time index, our goal is to estimate the unknown states associated with user's hand at each time step t, i.e., $\mathbf{x}_t = [x_t, y_t, x_{t-1}, y_{t-1}, \dot{x}_t, \dot{y}_t]^T$. x_t and y_t represent the coordinates of pixel for user's hand at time index t. $\dot{x}_t$ and $\dot{y}_t$ denote velocity at x and y directions at time index t.

In this paper, the state model describes the state relation from $t-1$ to t is defined as

$$\mathbf{x}_t = \mathbf{V} \cdot \mathbf{A} \cdot \mathbf{x}_{t-1} \ . \tag{8}$$

It is assumed that user's hand changes coordinates in each direction following a constant velocity model. State transition matrix $\mathbf{A}$ and process noise matrix $\mathbf{V}$ are defined as

$$\mathbf{A} = \begin{bmatrix} I_2 & \mathbf{0}_{2\times2} & I_2 \\ \mathbf{0}_{2\times2} & I_2 & \mathbf{0}_{2\times2} \\ I_2 & -I_2 & \mathbf{0}_{2\times2} \end{bmatrix}, \mathbf{V} = \begin{bmatrix} I_4 & \mathbf{0}_{4\times2} \\ \mathbf{0}_{2\times4} & \begin{bmatrix} v_{xt} & 0 \\ 0 & v_{yt} \end{bmatrix} \end{bmatrix} \ ,$$

I_m is a $m \times m$ identity matrix with diagonal term equal to 1 and others equal to 0. $\mathbf{0}_{m\times n}$ is a $m \times n$ zero matrix. v_{xt} and v_{yt} are independent Gaussian random variables as $\mathcal{N}(0, 1^2)$ which describe velocity is amplified or reduced by v_{xt} and v_{yt} from $t-1$ to t. The measurement model describes the relation between the captured frame and the user's hand as

$$\mathbf{z}_t = \sum_{\mathbf{x} \in R_{\mathbf{x}_t}} \rho(\mathbf{x}) + \mathbf{n}_t \ . \tag{9}$$

where

$$\rho(\mathbf{x}) = \begin{cases} 1 \ , \ T_{C_{r,l}} \leq C_r(x_t, y_t) \leq T_{C_{r,u}} \text{and} \\ \quad T_{C_{b,l}} \leq C_b(x_t, y_t) \leq T_{C_{b,u}} \\ 0 \ , \ \text{otherwise} \end{cases} \tag{10}$$

$R_{\mathbf{x}_t}$ is a predefined square region centered at $\mathbf{x}_t$. Instead of obtaining color from single pixel, all pixels in a square region are measured to decide if it is a hand pixel or not. $\mathbf{z}_t$ is the measurement which indicates if the pixel is skin-color or not. $\mathbf{n}_t$ is the measurement noise that affects color filtering.

In the proposed MARAIS, mobile view displays on the device at the beginning. After user touches the panel, user's hand tracking starts. $YCrCb$ color space is used in the measurement model. Y represents brightness information, while $C_r(x_t, y_t)$ and $C_b(x_t, y_t)$ represent Cr and C_b color information of pixel at (x_t, y_t). With the calibration process, the range of the Cr and Cb for hand color, i.e., $T_{C_{r,l}}$, $T_{C_{r,u}}$, $T_{C_{b,l}}$, and $T_{C_{b,u}}$ in (10), can be decided based on mean color (i.e., μ_{C_r} and μ_{C_b}) and variance of color (i.e., σ_{C_r} and σ_{C_b}) in a squared region. Note that each frame is filtered by considering only Cr and Cb space.

Particle filter [15] is a general Monte Carlo method for performing inference with state-space model at each time index. In PHANTOM, particle filter can deal with non-Gaussian measurement noise $\mathbf{n}_t$ in (9) and nonlinear state relation in (8). The idea is to form a weighted particle set $\{(\mathbf{x}_t^{(i)}, w_t^{(i)}), i = 1...N_s\}$ of the posterior distribution:

$$p(\mathbf{x}_t | \mathbf{z}_{1:t}) \approx \sum_{i=1}^{N_s} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}) \ , \tag{11}$$

where $\{\mathbf{x}_t^{(i)}, i = 1...N_s\}$ is a set of support points with associated weights $\{w_t^{(i)}, i = 1...N_s\}$ at time index t. Given initial distribution of estimate states $p(\mathbf{x}_0)$, prior distribution $p(\mathbf{x}_t | \mathbf{x}_{t-1})$ for state update, and likelihood $p(\mathbf{z}_t | \mathbf{x}_t)$ for measurement update, the posterior distribution can thus be calculated.

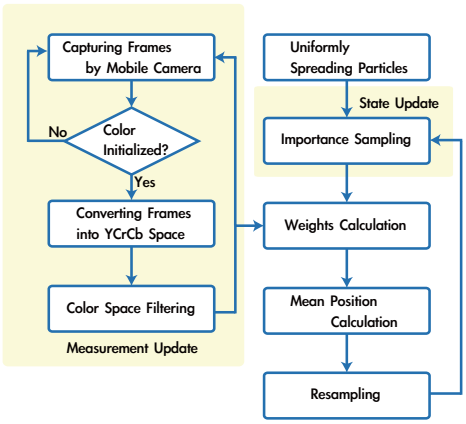


Fig. 3. Flow chart of PHANTOM.

The weights are calculated using the principle of sequential importance sampling particle filter [15]. If the samples $\mathbf{x}_{0:t}^{(i)}$ are drawn from an importance density $q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$, the weights in (11) is calculated as

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})}. \quad (12)$$

Sequential importance sampling chooses the importance density as the state transition probability, i.e., $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$. In (12), $w_t^{(i)}$ is further derived as

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot p(\mathbf{z}_t|\mathbf{x}_t^{(i)}). \quad (13)$$

Note that the likelihood for each particle $\mathbf{x}_t^{(i)}$ is calculated according the predefined region $R_{\mathbf{x}_t^{(i)}}$ centered at $\mathbf{x}_t^{(i)}$,

$$p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) = \frac{1}{N} \sum_{\mathbf{x} \in R_{\mathbf{x}_t^{(i)}}} \rho(\mathbf{x}), \quad (14)$$

where N represents the total pixel number in $R_{\mathbf{x}_t^{(i)}}$.

The idea of resampling is to reduce the number of particles with small weights and increase the number of particles with large weights to avoid potential degeneracy problem when the weights of particles are close to zero. After resampling, each particle's weight is adjusted to $1/N_s$ as the conventional resampling method. In other words, (13) becomes

$$\hat{w}_t^{(i)} \propto p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) \ , \quad (15)$$

and we can obtain $w_t^{(i)}$ after normalization. After updating weights of particles, we can estimate the location of user's hand in the sense of the minimum mean square error criterion as

$$E[\mathbf{x}_t|\mathbf{z}_{1:t}] \approx \sum_{i=1}^{N_s} w_t^{(i)} \mathbf{x}_t^{(i)}. \quad (16)$$

The flow chart of PHANTOM is depicted in Fig. 3. At each time step, measurement update and state update are performed following the above procedure. Note that there are skin-color objects in the background that may affect the tracking performance. Since user's hand color is measured at the initial calibration stage and hand motion is considered as a continuous movement in the state transition model, skin-color objects in the background can be removed from PHANTOM.

B. MARAIS at Cloud Side (MARAIS-C)

In Fig. 2, MARAIS-C executes feature detection and description for target image once when server startup to create a database that stores feature points of target image for matching. Through the network transmission, the system finds matching pairs between feature points of captured image from device and feature points of target image in database. Using these matching pairs, a homography matrix which transforms coordinates between two coordinate systems is calculated, i.e., a homography matrix for mobile display's coordinate system and target image's coordinate system. At last, MARAIS-C sends the corresponding four corner coordinates of the target image in mobile display coordinate system to MARAIS-D.

1) Feature Points Matching: The goal of this procedure is to find matching pairs between feature descriptors of database's target image and mobile view. Because MARAIS uses ORB feature extraction that generates descriptor in binary format, a binary matching algorithm is required for ORB descriptors. In this paper, Hamming distance is used to compare binary descriptors of captured frame and target image. The Hamming distance between two strings of equal length denotes the number of positions at which the corresponding bits are different. It can be calculated extremely fast on modern CPUs that often provide a specific instruction to perform a XOR or bit count operation. If a pair of feature descriptors have minimum Hamming distance below a specific threshold, the system considers them a matching pair.

(a) Homography matrix. (b) Homography matrix in our case.

Fig. 4. Meanings of homography matrix.

2) Finding Homography Matrix: As shown in Fig. 4(a), the meanings of homography matrix is a matrix that can transform one coordinate system to the other. In our case, we need to find a homography matrix that transforms coordinates of target image within the database to camera display's coordinates system as shown in Fig. 4(b). After obtaining matching pairs, we can find an optimal homography matrix by RANSAC [16], in which the system random selects matching pairs to calculate the homography matrix and determines the inliers. If there are sufficient numbers of the matching pairs which are regarded as inliers, the matrix is the optimal homography matrix. MARAIS-C transmits coordinates of target image in camera view obtained by the homograph matrix to MARAIS-D and then MARAIS-D will render the 3-D object according

to the coordinates information from MARAIS-C. Combining with PHANTOM, user can interact with the 3-D object by his own hand.

IV. Implementation Results

There are three parts in this section, including comparison of MARAIS and local MARAIS named MARAIS-L, performance validation of PHANTOM, and the demo of MARAIS. Note that MARAIS-L implements all the components in Fig. 2 at device side. In the paper, MARAIS-D is implemented on Samsung Galaxy Note N8010 with 1.4GHz quad-core CPU and the MARAIS-C is a 3.07GHz octa-core CPU server. The network environment is using WiFi for Internet connection between MARAIS-C and MARAIS-D. The measured download data rate is 11.05 Mbps the upload data rate is 13.3 Mbps in the tested environment. Note that the data rate can be higher since IEEE 802.11n supports maximum data rate up to 600 Mbps.

The target image is as shown in Fig. 4(b) and the frame size captured by the camera in the resolution of 1024×576 is about 1.8 MB. With feature extraction, the measured maximum upload data size from MARAIS-D to MARAIS-C for the transmission of detected feature points and its descriptor is 30 KB. The download data size from MARAIS-C to MARAIS-D for the transmission of boundary coordinates of target images is 128 bits.

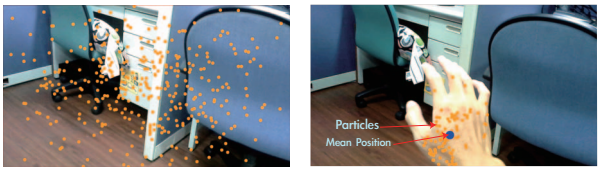
TABLE I
Comparison of MARAIS and MARAIS-L

	MARAIS	MASAIS-L
Initial Time (s)	1.3	2.6
Used Heap (%)	47	57
Detection Time (ms)	87.829	149
Description Time (ms)	139.5	128.6
Matching Time (ms)	10	11.639
Finding Homography (ms)	105	354.85
Processing Delay (ms) (Device Side)	227.329	644.089
Processing Delay (ms) (Cloud Side)	115	-
Upload Time (ms)	18	-
Download Time (ms)	0.011	-

A. Comparison of MARAIS and MARAIS-L

In this subsection, MARAIS and MARAIS-L are compared by the measured parameters of the whole process for a single frame processing. In Table I, the first row indicates the initial time of two systems when they startup. MARAIS reduces the waiting time for target image detection. The second row is the percentage of heap usage, which shows that cloud architecture reduces the memory loading about 10%. Note that processing delay of MARAIS-D contains only detection and description time while processing delay of MARAIS-C includes matching followed by finding homograph matrix. On the other hand, processing delay of MARAIS-L contains detection, description, matching and finding homography matrix at devices side. Therefore, finding homography matrix at cloud side saves many times.

Compared to cloud side, device side possesses limited computational power and memory. The proposed MARAIS is implemented using multi-threads where MARAIS-D can detect and describe current captured features in one thread. When the features are uploading to MARAIS-C, the saved computational power for homography matrix finding can be used to detect next frame in another thread. There is no thread blocking problem in MARAIS by offloading heavy tasks to clouds. The frame per second (FPS) which indicates the captured frames rate is used as a measure of fluency. The average frame rate of MARAIS with two thread is 9.03 FPS while that of MARAIS-L is 1.98 FPS, which indicates the differences on real-time processing abilities for two systems. In summary, the comparison that MARAIS is better in efficiency and more flexible than MARAIS-L validates the necessity of introducing cloud computing to MARAIS.



(a) Skin-color background. (b) Particle-based hand tracking.

Fig. 5. Results of PHANTOM.

B. Results of Particle-based Hand Tracking

In this subsection, the total particle number of MARAIS, i.e., N_s, is set as 400. The predefined skin-color range based on the user's hand color calibration is set as two times of standard deviation of Cr and Cb centered at the mean of Cr and Cb in a 10 pixels by 10 pixels square region, e.g., $T_{Cr,l} = \mu_{Cr} - 2\sigma_{Cr}$. In Fig. 5, orange points are scattered particles in the tracking process and the blue point represents the estimated position of user's hand. As shown in Fig. 5(a) and Fig. 5(b), the interference of skin-color like objects is successfully reduced by the proposed PHANTOM.

In the experiment, user's hand moves in the camera's view in which the captured frames and correct hand positions at different time instants are recorded. In Fig. 6, three different methods are compared for the hand's position estimation with the captured frames. Note that cumulative distribution function (CDF) of estimation error is in the unit of pixel. Red curve denotes PHANTOM in MARAIS. Green curve refers to WHOle Picture-based hand Position EstimatoR (WHOPPER), which directly calculates each pixels for the entire captured frame and take the average of the coordinates of the hand-color pixels to represent the hand position. Blue curve denotes a simplified scheme for PHANTOM without velocity tracking named PHANTOM-S, i.e., $\mathbf{x}_t = [x_t, y_t]^T$ in (8).

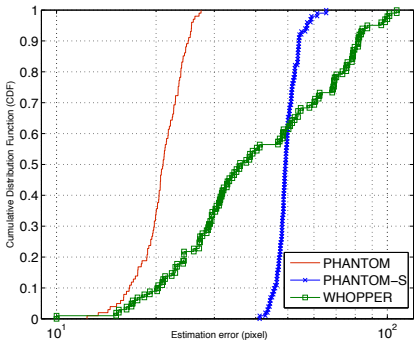
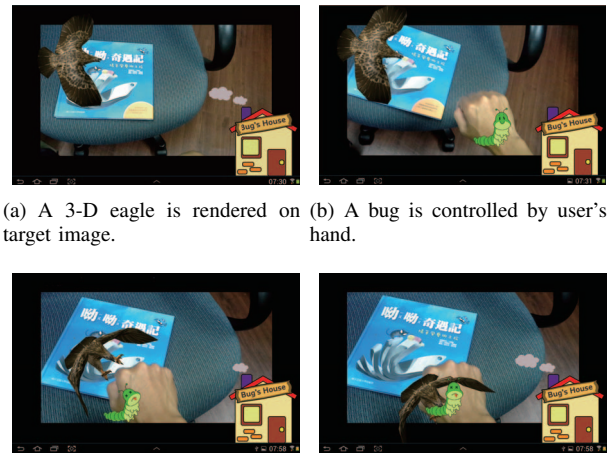


Fig. 6. Comparison of estimation error.

Note that the captured frame is with resolution of 1024×576. The computation load is high since 589,824 pixels have been computed for each frame for WHOPPER. By introducing state equation to describe user's hand, particle filter based method reduces computational complexity into $N_s = 400$ particles. Although WHOPPER keeps the whole pixels information, the performance would be interfered by skin-color objects in background. This demonstrates the necessity to represent the hand position in a mathematical model for more efficient computation. PHANTOM-S has the worst performance due to the lack of velocity information. Therefore, the proposed PHANTOM considers velocity information in its state which outperforms WHOPPER in performance while maintain lower computational complexity.



(a) A 3-D eagle is rendered on target image.

(b) A bug is controlled by user's hand.

(c) The bug moves close to the eagle.

(d) The bug is caught by the eagle.

Fig. 7. Application demo of MARAIS.

C. Demo of MARAIS

In this subsection, we implement MARAIS as an interactive application for digital learning system. The target image is the book cover in Fig. 7. When the target image appears in mobile camera view, the system renders a 3-D eagle model on the target as in Fig. 7(a). A bug lives in the house at the right lower corner and user can control the bug strolling around the mobile view by PHANTOM in Fig. 7(b). If the bug moves close to the eagle in Fig. 7(c), it would be caught by the eagle and lose its life in Fig. 7(d).

V. Conclusion

The proposed Mobile Augmented ReAlity Interactive System based on cloud computing (MARAIS) integrates cloud computing, augmented reality (AR), and Particle-based HANd Tracking methOd on Mobile (PHANTOM). It divides the whole AR process into two parts, including MARAIS at device side (MARAIS-D) and MARAIS at cloud side (MARAIS-C), to ease the computational burden of mobile platforms. The heavy task is performed at MARAIS-C, and the MARAIS-D is responsible for the sensing and detecting. The implement results show that the device/cloud architecture significantly improves system frame rate. On the other hand, the proposed PHANTOM filters out the skin-color pixels occasionally appearing in the background in order to provide feasible tracking of user's hand position. As the demo example, the proposed MARAIS can allow user to read a picture book by camera of mobile devices and interact with the projected 3-D object, which improves the user experience (UX) for digital learning.

References

[1] J. Bolter and B. Macintyre, "Is It Live or Is It AR?" *IEEE Spectrum*, vol. 44, no. 8, pp. 30–35, Nov. 2007.
[2] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.
[3] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
[4] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.
[5] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *IEEE Transactions on, Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.
[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.
[7] R. Y. Wang and J. Popović, "Real-time Hand-tracking with a Color Glove," *ACM Transactions on Graphics*, vol. 28, no. 3, 2009.
[8] S. Malik, "Real-time Hand Tracking and Finger Tracking for Interaction," Department of Computer Science, University of Toronto, Tech. Rep., 2003.
[9] C.-P. Chen, Y.-T. Chen, P.-H. Lee, Y.-P. Tsai, and S. Lei, "Real-time Hand Tracking on Depth Images," in *Proc. of IEEE Visual Communications and Image Processing (VCIP)*, Nov. 2011, pp. 1–4.
[10] B.-R. Huang, C. H. Lin, and C.-H. Lee, "Mobile Augmented Reality Based on Cloud Ccomputing," in *IEEE International Conference on, Anti-Counterfeiting, Security and Identification (ASID)*, 2012, pp. 1–5.
[11] Qualcomm, "Vuforia Developer," https://developer.vuforia.com/.
[12] OpenCV, http://opencv.org/.
[13] OpenGL, http://www.opengl.org/.
[14] P. L. Rosin, "Measuring Corner Properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999.
[15] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.
[16] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.