IEIE Transactions on Smart Processing and Computing, vol. 3, no. 2, April 2014
http://dx.doi.org/10.5573/IEIESPC.2014.3.2.88

Photo-based Desktop Virtual Reality System Implemented on a Web-browser

Masaya Ohta[1], Hiroki Otani[1], and Katsumi Yamashita[1]

[1] Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho Nakaku Sakai Osaka, 599-8531, Japan
ota@eis.osakafu-u.ac.jp

***** Corresponding Author: Masaya Ohta

Received November 20, 2013; Revised December 27, 2013; Accepted February 12, 2014; Published April 30, 2014

** Extended from a conference: Preliminary results of this paper were presented at the ICEIC 2014.This present paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

** Regular Paper*

***Abstract*:** This paper proposes a novel desktop virtual reality system. Based on the position of the user's face, the proposed system selects the most appropriate image of an object from a set of photographs taken at various angles, and simply "pastes" it onto the display at the appropriate location and scale. Using this system, the users can intuitively feel the presence of the object.

***Keywords*:** Virtual reality, Desktop VR, Photo AR, E-Commerce, Digital signage, HTML5/JavaScript

1. Introduction

Virtual Reality (VR) has been evolving constantly since its early days, and is now a fundamental technology in different application areas, including scientific and medical visualization, entertainment, video games, education, and training. One of the aims of this study was to apply VR technology to the display of products on electronic commerce (EC) sites and digital signage.

Object VR [1] is one of the notable methods that use the photographs of an object taken at various angles and allows the users to observe it from the angle they desire. On the other hand, Object VR requires dragging the mouse of a computer to change the angle. Therefore, the user does not feel the presence of the object in the display during mouse operation.

Previously, a photo-based augmented reality (Photo AR) system that uses the photo images of an object captured at various angles instead of rendering a 3D model corresponding to the object during runtime was proposed [2]. With this system, an appropriate photo image for the position and orientation of the user's camera was selected from previously captured and stored images, and then adjusted and simply "pasted" into the camera view. Because Photo AR displays the object as if it exists right in front of the user, it allows the user to intuitively recognize the object better than Object VR. On the other hand, the user needs to move the camera or marker when he/she wants to check the object at a different angle. Although this operation is more intuitive than dragging the mouse, it is not effective enough.

Desktop VR, which is commonly referred to as fish tank VR, is a technology that involves the use of a display of a desktop computer coupled with a head tracker that estimates the user's head position and updates a 3D projection matrix in real-time [3-5]. This system allows the users to observe virtual images through the display as if they were looking into an actual fish tank. He/she feels the presence of a rendered object in the display because no mouse operation is needed. Although, desktop VR does not provide strong immersion, it is suitable for visualization systems because of its ease of use and ability to present high-quality images.

The main obstacle encountered when applying desktop VR to EC sites is that 3D models of all viewable products must be prepared in advance. A typical desktop VR system uses three dimensional computer graphics (3DCG) to render the objects required for a given user's viewpoint. These objects are typically authored using 3D modeling programs. These tools are highly expressive but they also tend to be complex and time-consuming. If the number of viewable products is large, the cost of creating these models will in most cases be prohibitive. 3D reconstruction methods have been proposed [6-8] as an alternative to

author-driven modeling. These methods attempt to reconstruct a 3D model, automatically or semi-automatically, based on the photo images of a target object. On the other hand, accurate modeling may be impossible in some cases when a target object has a complicated shape. This technology is inappropriate because the product images displayed on EC sites must be accurate and appealing.

This paper proposes a photo-based desktop VR system that uses the photo images of an object captured at various angles, instead of rendering a 3D model of the object during runtime. In this system, the appropriate photo image for the position of the user's face is selected from the images captured and stored previously, which is then adjusted and simply "pasted" onto the display. By using this system, the users can strongly feel the presence of the rendered object. This system does not use 3DCG, and can display the products accurately. This approach can reduce the number of rendering calculations significantly because it does not require 3DCG rendering. Moreover, any complicated image can be processed at the same processing load because photo images are just pasted. This also spares the developers from the efforts of creating a 3D model for every viewable object. In addition, this helps to minimize the costs of rendering on mobile devices with limited battery capacity, such as smart phones. The system is implemented using HTML5 and JavaScript; hence, it has little dependence on the OS or browser, which can reduce the development costs.

The remainder of this paper is organized as follows. Section 2 provides an outline of the system and Section 3 presents an evaluation of the proposed methods. The final section summarizes this study.

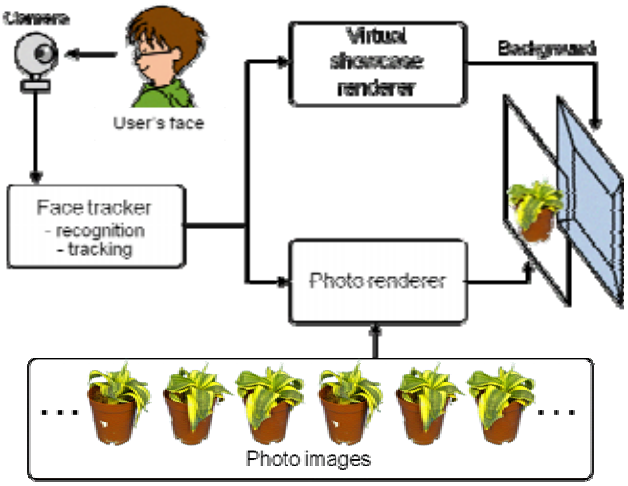

Fig. 1. Configuration of the proposed system.

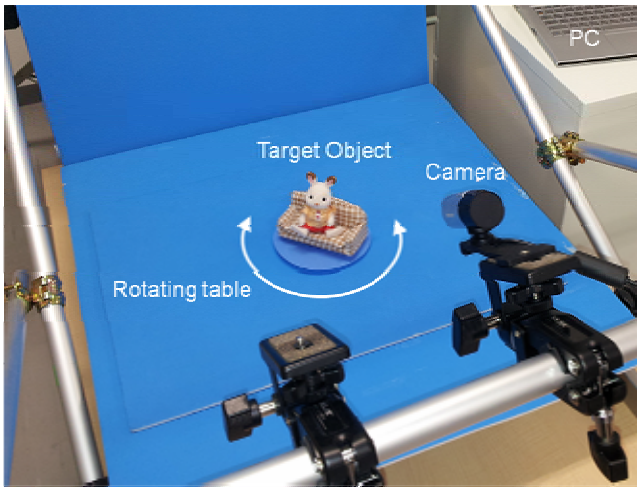

Fig. 2. Imaging system.

2. Proposed System

2.1 Configuration of the Proposed System

The desktop VR system proposed in this study consists of a face tracker, a photo renderer, and a virtual showcase renderer. Fig. 1 presents the configuration of the proposed system. The face tracker recognizes the user's face at the image captured by a camera and estimates the position relative to the display.

The photo renderer receives the results of the face tracker, selects a suitable photo image from a set of images captured and stored previously, and draws an image on the display. In addition, the virtual showcase renderer receives the results of the face tracker, and draws the virtual showcase such that the user feels the target object being displayed in the showcase.

The face tracker, the photo renderer, and the showcase renderer were implemented using HTML5 and JavaScript and run on a web browser with an HTML rendering engine.

2.2 Photo Images

This section describes the photography method implemented to capture the photographs of the products used for the system. Fig. 2 shows a simple imaging system. First, the target object is photographed by the imaging system, which is the same as that for Object VR. In this system, the object is placed on a rotating table and photographed from a number of preset angles using a camera. These initial images are stored in video format. To remove the background from the images, blue-screen panels are placed around the object so that their color can be deleted from the resulting images using video editing software. The final images are stored as a png format file with an alpha channel.

Because the proposed system requires a transparent background, the photo images are stored in png format, as mentioned above. The png format has an alpha channel in addition to RGB, and is suitable for images with a transparent background. For this purpose, however, it must handle a large volume of data. On the other hand, the data volume of the jpeg format is small because it does not store images with a transparent background due to the lack of an alpha channel. Therefore, to reduce the amount of data, the png images are converted to jpeg using the system, and a transparent background is created when drawing the

images. In particular, information of the transparent areas is stored separately as data similar to the run-length coding, in addition to the jpeg images. The system uses this information to draw pixels consisting of only non-transparent areas on a background image to create images with a transparent background.

2.3 Face Tracker

The face tracking system used in this section is an open source system presented in reference [9]. This system was implemented with HTML5/JavaScript, and was executable on browsers. The face tracking system consisted of two phases. First, the recognition of a face captured by a camera is attempted. This is called the recognition phase. Second, the face is tracked using a tracking method. This is referred to as the tracking phase. If the tracking phase fails, it returns to the recognition phase. The recognition and tracking results are calculated with the display coordinate system, as shown in Fig. 3.

2.4 Photo Renderer

Suppose the coordinates of a user's face calculated by the face tracker are (X,Y,Z), the Photo renderer calculates the angle formed by the line-of-sight from the center of the display to the face center and the Y axis of the display coordinate system is specified as the depression φ, and the angle formed by the straight line as the projection of this line-of-sight onto the ZX plane of the display coordinate system and the Z axis is specified as the azimuth ψ (See Fig. 3). φ and ψ are calculated as follows.

$$\varphi = \tan^{-1}\left(\sqrt{X^2+Z^2}\,/\,|Y|\right) \quad (1)$$

$$\psi = -\tan^{-1}(X/Z) \quad (2)$$

In this system, the image shot from a position closest to both φ and ψ is selected. The image it then is pasted on the proper position of the display.

2.5 Showcase Renderer

To indicate a showcase that is located at the back of the display and that contains a product, a virtual box is drawn in the background of the product. Fig. 4 shows the drawing method of the showcase. When the depth of the virtual showcase is D, the intersection point (x_i, y_i) between the display and a straight line connecting $(X_i, Y_i, -D)$ and (X,Y,Z) can be calculated using the following formula:

$$x_i = X - \frac{Z}{D+Z}(X - X_i) \quad (3)$$

$$y_i = Y - \frac{Z}{D+Z}(Y - Y_i) \quad (4)$$

The lines connecting the intersection points and the four corners of the display are drawn as a virtual showcase.

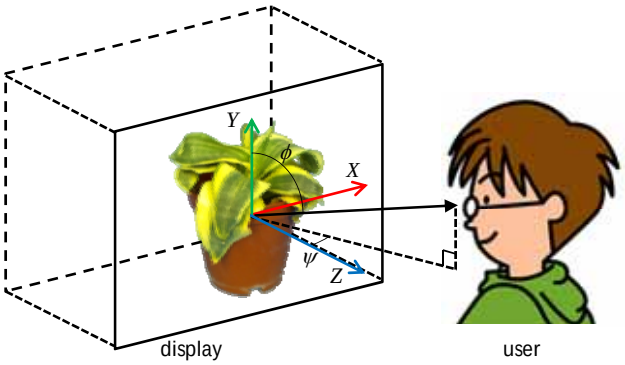

Fig. 3. Display coordinate system.

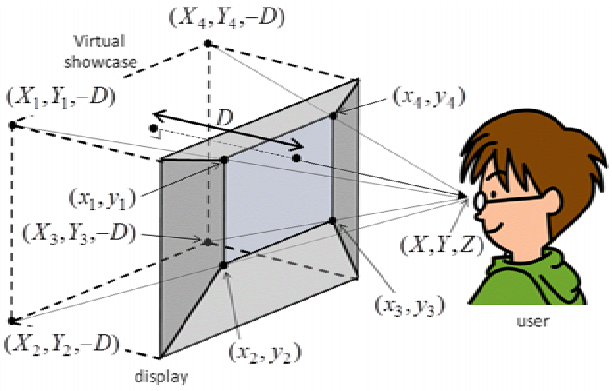

Fig. 4. Rendering of a virtual showcase.

Fig. 5. Captured images for face tracking (left: desktop PC, right: tablet).

3. Evaluation

3.1 Performance of Face Tracker

The performance of Face tracker was evaluated experimentally. A desktop PC and tablets (platforms) shown in Table 1 were used for the experiments.

First, to evaluate the performance of only the recognition phase, the system was modified to skip the tracking phase every time, and the number of times of recognition processing executed per second was measured. During the measurement, the face was kept still. Moreover, a partition was placed at the back of the user to create a completely white background, so the recognition experiment could be performed in every platform under the same conditions.

Fig. 5 shows the images captured during the measurement, and Fig. 6 shows the measurement results.

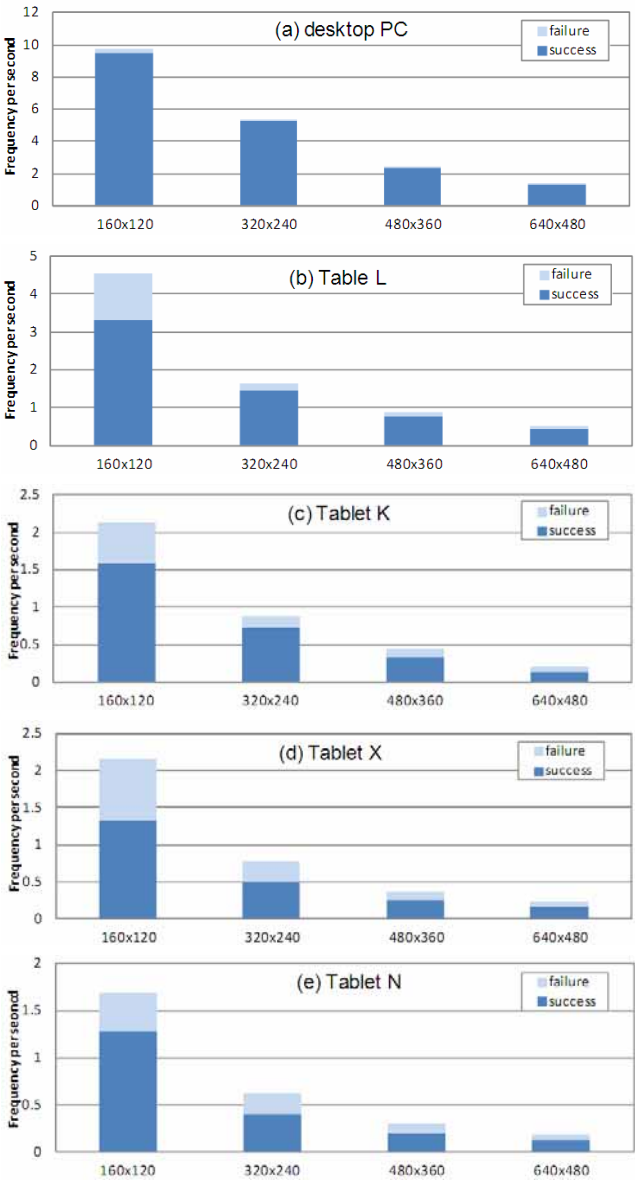

Fig. 6. Performance of face recognition.

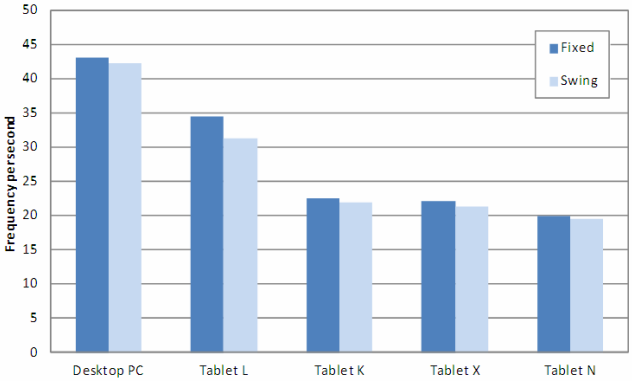

Fig. 7. Face tracking performance.

The dark blue bars show the number of successful recognitions, and the pale blue bars show the number of times of failed recognitions. These graphs indicate that additional time is required for recognition as the resolution of images captured by the camera increased. When performing the following experiments, the resolution of the image for which the recognition time was shortest should be used.

Subsequently, the tracking phase performance was evaluated. In addition, the number of times tracking the face was executed every second for two cases was evaluated: the face was static, and the face swung right and left (approx. 30cm/sec).

Fig. 7 shows the measurement results. Compared to the static face, the swinging face needed at most 1.1 times longer to track a face. In addition, compared to Fig. 6, the tracking processing was 4.5 times or significantly faster than that of recognition processing. No case of failure was noted when the face was swung just around 30cm/sec.

3.2 System Performance

The proposed system was implemented with HTML5/JavaScript, and the performance was evaluated. First, an operation test was conducted with the devices

Table 1. Specifications of the desktop PC and tablets for the experiments.

Platform	Desktop PC	Tablet L	Tablet K	Tablet X	Tablet N
Maker/Type	Custom PC	Lenovo/ ideaPad Miix2 8	Amazon/ Kindle Fire HDX 7	Sony/Xperia Z SGP412JP/W	Google/ Nexus 7 (2013)
CPU	Intel Core i5-3470 3.2GHz	Intel Atom Z3740	Qualcomm Krait 400x4	Qualcomm Krait 400x4	Qualcomm Quad Krait
RAM	8GB	2GB	2GB	2GB	2GB
GPU	NVIDIA GeForce GTX 650	Intel HD Graphics	Adreno 330	Adreno 330	Adreno 320
Monitor	24inch 1,920x1,080	8 inch 1,280x800	7 inch 1,920x1,200	6.4 inch 1,920x1,080	7 inch 1,920x1,200
Camera	3MPixel	2MPixel	1.2MPixel	2.2MPixel	1.2MPixel
OS	Windows 8.1	Windows 8.1	Fire OS 3.0	Android 4.2	Android 4.3
Browser	Google Chrome 33.0.1750.154m	Google Chrome 33.0.1750.154m	Google Chrome 33.0.1750.136	Google Chrome 33.0.1750.166	Google Chrome 33.0.1750.166

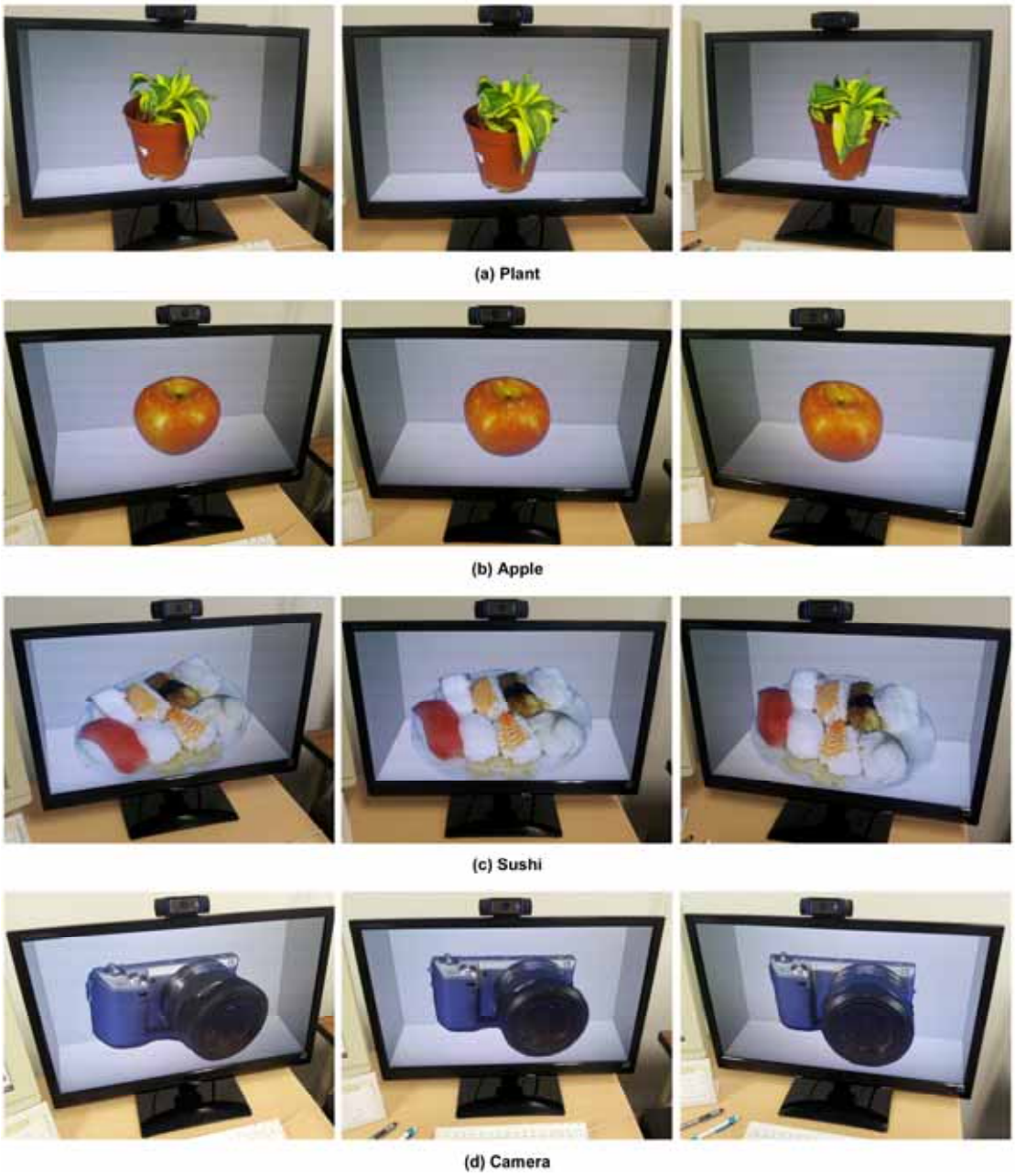

Fig. 8. Display of the proposed system (desktop PC).

shown in Table 1. One hundred and twenty pictures photographed at every 3° from 360° around a product were prepared. The resolution of each photo image was 720x720 pixels. In this experiment, a face that rarely moved up and down was considered, the angle of depression (φ) was fixed at 60°.

All photo images were saved in the desktop PC used for the experiment. When this PC was used for the experiments, the browser on the PC read these images directly. When using a tablet for the experiment, the desktop PC was used as a Web server, and the system was prepared so the tablet could download the photo images from the server as needed.

The system was also prepared using 3DCG instead of photo images, and compared with the proposed system that used the photo images. The open source library, "Three.js" was used for 3DCG rendering [10].

Figs. 8 and 9 show the operation checking process. As shown in the figures, images could be rendered smoothly on all platforms, and the test users felt as if the product really existed in the showcase installed in the display. Fig. 10 shows a display of the system using 3DCG. A comparison of Figs. 8(a) and (b) with Fig. 10 showed that the proposed system using the photo images could render images more realistically than the 3DCG system, and it was suitable for displaying products.

Subsequently, the processing time of the Photo renderer was measured as the face swung right and left. The size and the number of photo images were the same as those of the previous experiment.

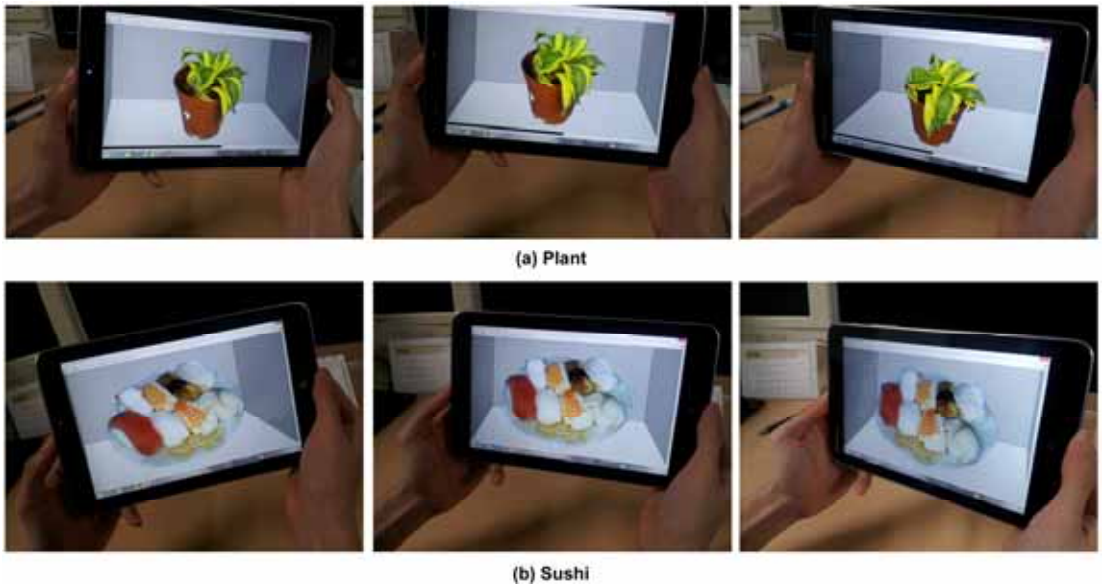

Fig. 9. Display of the proposed system (Tablets).

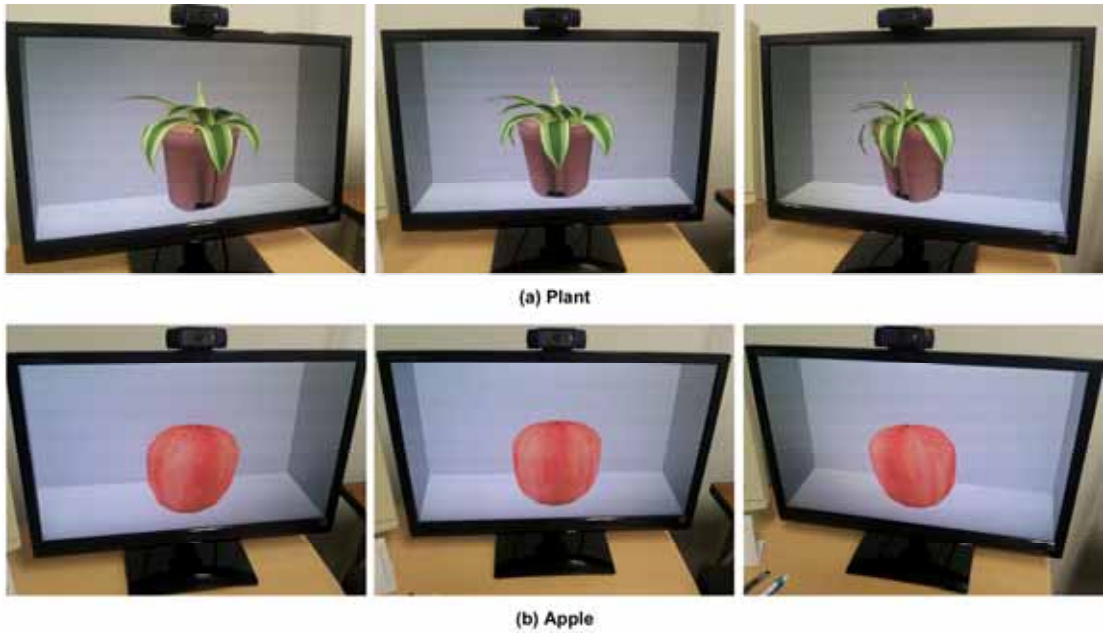

Fig. 10. Display of the system using 3DCG (desktop PC).

Fig. 11 shows the measurement results. The vertical axis of the figure shows the inverse number of measurement results and the means of how many times per second the Photo renderer could render. As shown in the figure, in every platform, the calculation load of the processing by the Photo renderer was smaller than that for recognition processing by Face Tracker.

Next, the processing time of the conventional 3DCG renderer (canvas renderer of Three.js) and the proposed Photo renderer were compared. The 3D model "Apple" consisted of 208 vertices and 228 faces. The 3D model "Plant" consisted of 10116 vertices and 9952 faces. Similar to the previous experiment, the face was swung right and left during the measurements.

Fig. 12 shows the measurement results. For the model "Apple", the conventional 3DCG renderer was faster than the Photo renderer because the size of the model is very small. On the other hand, for the model "Plant", the processing time could not be measured on all tablets due to the large size of the model. In contrast, photo renderer could operate at almost same speed regardless of the complexity of the model.

Finally, the resolution of the photo images was set to 1080x1080 pixels, and the number of photos was also increased from 120 pictures (every 3°) to 500 pictures (every 0.72°) so that the product could be indicated smoothly according to movement of the user' face, and the processing time of Photo renderer was measured. the number of different photo images displayed per second was also measured.

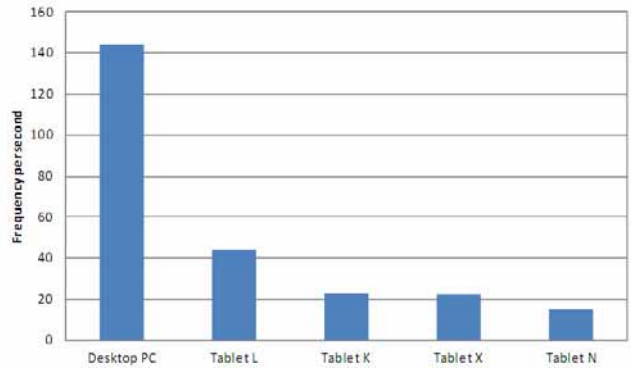

Fig. 11. Performance of the photo renderer.

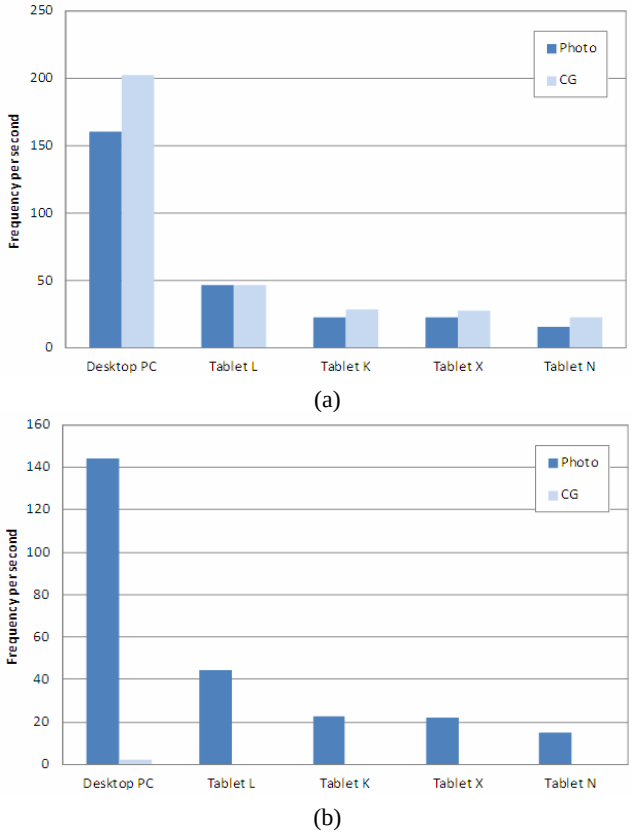

Fig. 12. Comparison of the rendering performance (CG: conventional, Photo: proposed).

Fig. 13 shows the measurement results. Although the number of resolution or photo images was increased, the system on the desktop PC and Tablet L could draw images at 20 fps or higher. Fig. 14 shows the number of photo images displayed per second. The figure shows that smoother images were displayed as the number of displayed different photo images was increased. When checking the operation visually, no switching of the photos could be found and very natural images were observed.

3. Conclusion

This paper presented a novel desktop VR system. The proposed system was implemented and evaluated on a desktop PC and tablets. From the evaluation, it was verified that this system operated smoothly on both a PC and tablets, and could provide a better sense of its existence to users.

Recently, LCD displays have become increasingly larger in size and their resolutions have increased proportionally. These devices are suitable for displaying products accurately and beautifully. We believe that this system will be able to display products more attractively when incorporated in these devices.

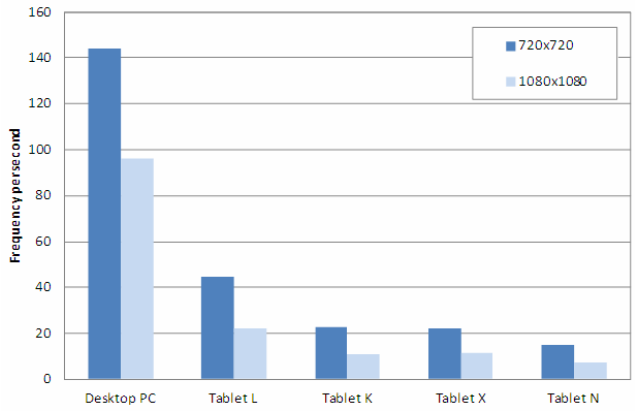

Fig. 13. Performance of the photo renderer with high-resolution photo images.

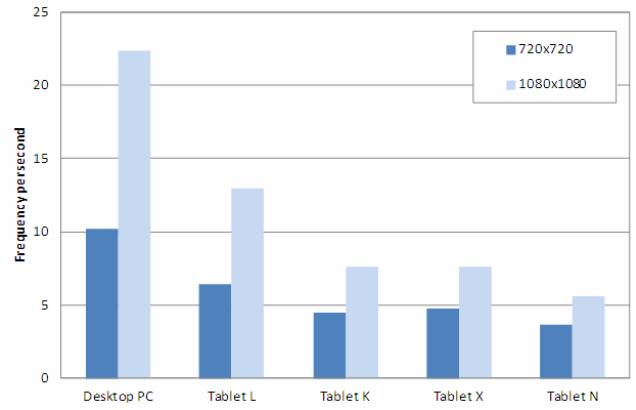

Fig. 14. Number of different photo images displayed per second.

References

[1] T. Monroe, QuickTime Toolkit: Advanced Movie Playback and Media Types, Morgan Kaufmann, 2004, pp. 283-287. Article (CrossRef Link)

[2] M. Ohta, R. Yokomichi, M. Motokurumada, K. Yamashita, " A Photo-Based Augmented Reality System with HTML5/JavaScript," Proc. of 1st IEEE Global Conf. on Consumer Electronics, pp.430-431, 2012. Article (CrossRef Link)

[3] K. W. Arthur, K. S. Booth, and C. Ware, “Evaluating 3D task performance for fish tank virtual worlds,” ACM Transactions on Information Systems, vol.11,

no.3, pp.239-265, 1993. Article (CrossRef Link)
[4] J. Rekimoto, “A vision-based head tracker for fish tank virtual reality,” Proc. of Virtual Reality Annual International Symposium, pp.94-100, 1995. Article (CrossRef Link)
[5] J. C. Lee, “Hacking the Nintendo Wii Remote,” IEEE Pervasive Computing, vol.7, no.3, pp.39-45, 2008. Article (CrossRef Link)
[6] M. Brown, T. Drummond, and R. Cipolla, “3D Model Acquisition by Tracking 2D Wireframes,” Proc. of the 11th British Machine Vision Conference, 2000. Article (CrossRef Link)
[7] A. van den Hengel, A. Dick, T. Thormahlen, B. Ward, and P. H. S. Torr, "Video-Trace: Rapid Interactive Scene Modelling from Video,” ACM Transactions on Graphics, vol. 26 (3), 2007. Article (CrossRef Link)
[8] Q. Pan, G. Reitmayr, and T. Drummond, “ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition,” Proc. of the 20th Brithish Machine Vision Conference, 2009. Article (CrossRef Link)
[9] headtrackr. Article (CrossRef Link)
[10] Three.js. Article (CrossRef Link)

Masaya Ohta graduated from Osaka Prefecture University in 1991, completed his doctoral studies in 1996, and became an assistant professor at Osaka Electro-Communication University, an assistant professor with the Graduate School of Engineering at Osaka Prefecture University in 2002. He has been an associate professor since 2012. He is primarily pursuing research related to augmented reality and communications systems. He holds a D.Eng. degree, and is a member of IEEE, IEICE, IEEJ, and IPSJ.

Hiroki Otani graduated from Osaka Prefecture University in 2012, and enrolled in the first half of the doctoral program there. He is primarily pursuing research related to augmented reality and is a member of IPSJ.

Katsumi Yamashita completed his doctoral studies at Osaka Prefecture University in 1980. He became a lecturer on the Faculty of Engineering at the University of the Ryukyus in 1982, an associate professor in 1988, and a professor in 1991. Since 2000 he has been a professor in the Graduate School of Engineering at Osaka Prefecture University. He has primarily been pursuing research on communications and adaptive signal processing. He holds a D.Eng. degree, and is a member of IEEE and IEEJ.