Real-Time Blur Rendering of Moving Objects in an Augmented Reality Environment

Zhao Yue
School of Information Science & Engineering
Northeastern University
Shenyang, China
zy_ky7777@126.com

Li Jing-jiao
School of Information Science & Engineering
Northeastern University
Shenyang, China

Li Zhen-ni
School of Information Science & Engineering
Northeastern University
Shenyang, China

Ma Ji
School of Information Science & Engineering
Northeastern University
Shenyang, China

Abstract—**When moving objects occur a large number of blurred situations, the augmented reality rendering virtual objects can't consistent with motion blur. As the above problem, we proposed a three-dimensional motion blur rendering virtual objects algorithm. First, an image structure model of motion-blurred template matching was used, which was used to track motion-blurred objects. Second, we used the same image structure model to generate blur. We produced motion blur for 3D motion by mixing 2D distorted and 3D rendering. Through experimental comparison, our method generated virtual blurred objects realistically. And our rendering speed was faster than without using OpenGL approximation. Therefore, our method was better performance.**

Keywords-3d motion blur target; blur rendering; image structure model; augmented reality; intermediate image

I. INTRODUCTION

Augmented Reality (referred to as AR) is a technology which is use of computer system to produce three-dimensional information to increase user' perception of the real world [1]. With the rapid development of the webcam and mobile devices based on the AR applications, motion blur is a major obstacle to vision-based tracking and augmented reality [2,3].

When the motion blur is serious, rendering the virtual object must reflect the correct motion blur. Otherwise, they will look unnatural to the augmented reality application. In recent years, scholars have studied a number of rendering blur approach. Okumura et al considered only as lens blur caused an estimated point spread function of the image. It would be applied to the rendered image [4]. Fischer et al studied the two-dimensional translational motion blur. It simulated motion blur through translating the target image several times and summing the images which was created [5]. Klein and Murray recently proposed a method for generating motion blur due to the camera rotation [6]. In the process of tracking, they estimate image motion for the two-dimensional transformation and the combination of two-dimensional rotation. They made the virtual target had motion blur through convert virtual target with this movement. However, these methods were applied blurred artificial 2D image. They were restricted to a uniform 2D motion blur. The motion blurred quantity caused by the rotating is uneven, so it often failed to present a consistent blur effect.

To solve these problems, we propose a method that renders blur virtual objects under three-dimensional motion. First, an image formation model of motion-blurred template matching was used, which was used to track motion blur target. Second, we used the same Image formation model to generate blur. We produce motion blur for 3D motion by mixing 2D distorted and 3D rendering. Our algorithm avoids to produce a lot of duplication of rendering. At the same time, the rendered image blur and motion blurred object are basically the consistent.

II. IMAGE STRUCTURE MODEL

Firstly, we seriously consider moving objects which exist motion blur. Typically, the tracking method based on template matching algorithm assumes a simple image structure model $P(\varphi)$ [7]. Wherein, we transform a known template T to produce the captured image P_c :

$$P(\varphi) = \upsilon(T, \varphi) \tag{1}$$

Where, $\upsilon(.,.)$ is a distortion function. $\upsilon(.,.)$ and motion parameters φ jointly apply to the template T. The objective is to estimate the motion $\hat{\varphi}$ .We estimate $\hat{\varphi}$ from the correlation between the maximizing captured image P_c and the distorted template:

$$\hat{\varphi} = \arg\min_{\varphi} \left\| P_c - P(\varphi) \right\|^2 \tag{2}$$

However, the standard model fails to consider the possibility of motion blur. We generalize the model as:

$$P(s, t_0) = \frac{1}{1 - t_0} \int_{t_0}^{1} \upsilon(T, s(t)) dt \tag{3}$$

978-1-4799-4955-7/14 $31.00 © 2014 IEEE
DOI 10.1109/IHMSC.2014.136

That is, we assume that the template moving average distorted image under a motion defined by s(t) is the captured image. Where, t_0 is the shutter opening time, and can be normalized to between 0 and 1. Now, we expect the template is a full motion which is between t_0 and 1.

Simply put, we do assume a linear motion, so it can be said:

$$s(t) = t \cdot \varphi \tag{4}$$

Where, φ is the template movement when shutter close. We further simplify our image structure model (3) as:

$$P(\varphi, t_0) = \frac{1}{1-t_0}\int_{t_0}^{1} \upsilon(T, t\varphi)dt \tag{5}$$

The model for the motion blur is effective, but also to meet the case there is no motion blur. When $t_0 \to 1$, that is when we assume the capture is instantaneous, then the model is more close to the standard model (1).

III. 3D MOTION BLUR RENDERING

A. Blur Production

A general method to simulate motion blur is a virtual object along its path in the image exposure render multiple times and fuse these rendered images together. However, in order to create realistic smooth blur, such images require dozens of pieces. It is impossible for real-time 3D rendering application. In order to achieve blur effect to render model many times so that the system had to spend more computing time. It is unable to meet the real needs. In addition, computer vision algorithms are tending to use the GPU, thus leaving less time to render images [11].

Therefore, as described in Figure 1, our strategy is to simply render several images of virtual objects and by twisting rendered images to generate an intermediate image. We call these rare rendered images inner-frame. Since distortion is much faster than 3D rendering, we can correctly produce blurred images in very short time.

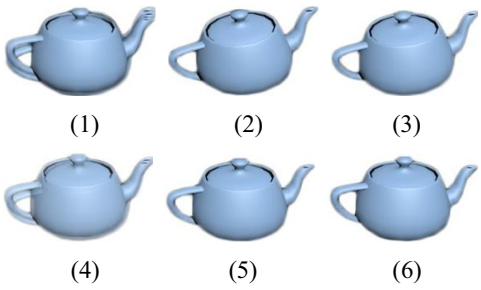


Figure 1. Our method of fuzzy rendering

Fig. 1 summarizes our effective method of blur rendering. In order to the rendered virtual object is consistent with the compensation motion, we firstly use OpenGL to generate two intermediate images (c) and (e) we call inner-frames. Then we create other intermediate images. We generate intermediate images by distorting rendered images. The two inner-frames are weighted and summed to obtain intermediate images such as (d).We respectively distort (c) and (e) to obtain (b) and (f).Lastly, we obtain the blurred image as (a) by summing all the obtained intermediate images.

More formally, we want to generate a virtual image P_{fic} can be written as:

$$P_{fic} = \frac{1}{1-t_0}\int_{t_0}^{1} r(o, s(t))dt, \tag{6}$$

Where $r(o, s(t))$ is the rendered image of the virtual object O at the position s(t). As discussed in Section 2, t_0 represents the camera shutter's open time, which is normalized to between 0 and 1. In addition, also in Section 2, we assume that s (t) linearly evolves between t_0 and 1. Therefore, the image structural model for rendering is similar to the image structural model for tracking as defined in second portion. However, we cannot assume that the goal pose is always small. We calculate s (t) when $s(t) = \varphi_a + t\varphi$. Where, φ_a is the object pose estimation which is compared to the previous captured image. φ is the object motion for the current image which is calculated in the method of the previous section.

In the experiment, we must replace the integral in (6) by a limited sum:

$$P_{fic} \approx \frac{1}{n+1}\sum_{i=0}^{n} r(o, \varphi_a + \frac{i}{n}\varphi). \tag{7}$$

The blur becomes more real with the increase of n. And the rendering time also increases.

As discussed above, in fact, we only use two inner-frames to render image into 3D. When i = n / 4 and i = 3n / 4, $|n/4 - 0| = |n/4 - n/2| = |3n/4 - n/2| = |3n/4 - n|$, the other images from the rendering of two inner-frames are the most homogeneous. Therefore, P_{fic} is approximated as:

$$P_{fic} \approx \frac{1}{n+1}\left(P_{\frac{n}{4}} + P_{\frac{3n}{4}} + \sum_{\substack{i=0 \\ i\neq\frac{n}{4}, i\neq\frac{3n}{4}}}^{n} Q_i \right) \tag{8}$$

Where $P_{\frac{n}{4}} = r(o, \varphi_a + \frac{\varphi}{4})$, $P_{\frac{3n}{4}} = r(o, \varphi_a + \frac{3\varphi}{4})$. Images Q_i are the intermediate images which are created for the other values of i from $P_{\frac{n}{4}}$ to $P_{\frac{3n}{4}}$.

B. Intermediate Image

We generate n intermediate images by affine transform from two inner-frames. One strategy is that only use the first inner-frame to generate the first portion of the intermediate images and use the second inner-frame to generate the second portion. However, there will be some incoherent in the connection. Thus, near the connection we fusion the distorted images from two inner-frames to avoid such discontinuities.

More accurate intermediate images Q_i are calculated as follows:

$$Q_i = \begin{cases} \omega(P_{\frac{n}{4}}, B_{\frac{n}{4}\to i}), & if \quad i < \frac{n}{4}, \\ \alpha(i)\omega(P_{\frac{n}{4}}, B_{\frac{n}{4}\to i}) + \beta(i)\omega(P_{\frac{3n}{4}}, B_{\frac{3n}{4}\to i}), & if \quad \frac{n}{4} < i < \frac{3n}{4}, \\ \omega(P_{\frac{3n}{4}}, B_{\frac{3n}{4}\to i}), & if \quad i > \frac{3n}{4}, \end{cases} \tag{9}$$

Where, when i changes from n / 4 to 3n / 4, $\alpha(i)$ is a linear transformation from 1 to 0. When i changes from n / 4 to 3n / 4, $\beta(i)$ is a linear transformation from 0 to 1. $\omega(.,.)$ is a transformation function and $B_{i\to j}$ is a affine transformation from the image i to the image j.

We only need to rely on the projection that from the corner of the border to the virtual object to calculate the affine transformation $B_{i\to j}$. From t_0 to 1, we estimate these corners of inner-frames and intermediate images with 3D transformation of the position. We calculate $B_{i\to j}$ by the affine transformation as best as possible in the least-squares sense. Therefore, it is approximately effective for any pixel on the virtual object.

We propose an adaptive scheme for determining the number of intermediate images for rendering. In order to obtain satisfactory rendering, small motion may require fewer images and large motion require more images. Therefore, we estimate the number of the intermediate images by the range of motion of the image. With the first protruding corner of the bounding, we estimate their position and calculate the distance from them to the previous images. Then, we use the average distance (in pixels) as the number of intermediate images. In practice, we limit the maximum number of 48, because it is acceptable visually in our experiment.

IV. EXPERIMENTAL RESULTS

We evaluated our motion blur rendering algorithm on a desktop computer with a 3.3GHz CPU, and Graphic Card NVIDIA GeForce GT 710M. The resolution of the camera used to track is 640 × 480P/30Hz. Its shutter speed could be fixed at the specified exposure time or automatically adjusted. In the experiment, the size of the template was set as 32 × 32 pixels. We rendered the virtual object in the same resolution in which the camera used to track is.

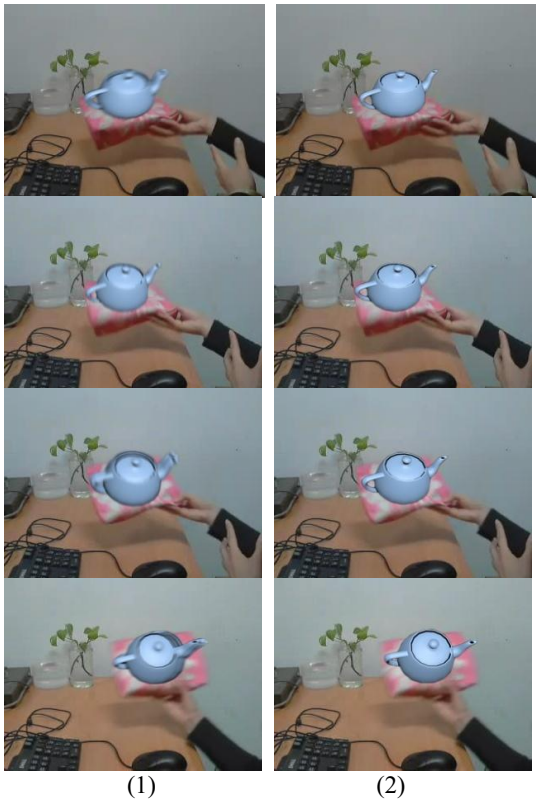
Experimental model was a teapot. The model was rendered as a display list had been compiled. The rendering was very fast, because all vertices and pixel data were stored in the graphics card.

The results of the generated motion blur which we proposed were shown in Fig. 2. Although we synthesized several variants of two-dimensional images to instead of several three-dimensional rendering images, the rendering image were almost without artifacts. It was almost unable to distinguish in real-time augmented reality.

In Fig. 2, the rendered blur with our method and the rendered blur with the other method were almost no difference, but our method was faster.

(1) (2)

Figure 2. Different algorithms produce motion blur results. (1)our method (2) Method of using 800 intermediate image rendering blur



(1) (2)

Figure 3. Motion blur generation and registration effect. (1)our method (2) rendering objects without motion blur

We use our method to render three-dimensional objects with a complex effect. We rendered virtual objects respectively in the different perspectives and the interpolation. Then we synthesized these two views to achieve the continuing validity. The test results on the image sequence cosmetic bag (length of 1235) shown in Fig. 3. The left column was rendered motion blur objects with our approach. The right column was rendered objects without motion blur. As can be seen from the figure 3, the embedded object was almost consistent with the three-dimensional object and it was lifelike in the left column. But the embedded object was very stiff in the right column.

The experimental object was the cosmetic bag sequence. Table 1 showed the average render time. Most of the objects rendering depend on the inner-frame rendering, because the object need to be drawn twice. Conversely, the intermediate images were rendered very easy. It was independent of the complexity of the object. We performed two operations on the graphics card. When GPU rendered enhanced image, CPU captured and tracked the next frame.

Table 2 showed the average render time on different methods.

TABLE I. AVERAGE TIMES OF FUZZY RENDERING PRODUCTION ON COSMETIC BAG SEQUENCES

Procedure	Average times (ms)
inner frames rendering	0.58
Intermediate image and blending	0.45
Final rendering	0.66

TABLE II. COMPARISON OF AVERAGE TIME OF DIFFERENT METHODS TO GENERATE FUZZY RENDERING ON COSMETIC BAG SEQUENCES

Method	Average times(ms)
Our method	1.69
Method of using 800 intermediate image rendering blur	15.32

V. CONCLUSIONS

In this paper, we proposed a new method. We produced motion blur for 3D motion by mixing 2D distortion and 3D rendering. By comparing the different methods, our method generated virtual objects blur realistically. Experimental results showed that the time that we use without OpenGL approximation generated virtual objects rendering is the time of our method almost 9.07 times. Therefore, the rendering speed of our method is faster.

REFERENCES

[1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. "Recent Advances in Augmented Reality," IEEE Computer Graphics and Applications, vol.21, pp.34-47, June 2001.

[2] Wu Yi, Ling Haibin, Yu Jingyi, Li Feng, Mei Xue, Cheng Erkang. Blurred Target Tracking by Blur-driven Tracker: Proceedings of the IEEE International Conference on Computer Vision, 2011[C]. Spain: IEEE, 2011:1100-1107.

[3] Jin H., Favaro P., Cipolla R. Visual Tracking in the Presence of Motion Blur: Proceedings. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005[C]. USA: IEEE Comput. Soc, 2005: 18-25.

[4] B. Okumura, M. Kanbara, and N. Yokoya. Augmented Reality Based on Estimation of Defocusing and Motion Blurring from Captured Images: Proceedings of Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality, 2006[C].USA:IEEE,2007:219-225.

[5] J. Fischer, D. Bartz, and W. Strasser. Enhanced Visual Realism by Incorporating Camera Image Effects: 2006 IEEE/ACM International Symposium on Mixed and Augmented Reality ,2006[C].USA: IEEE ,2006:205-208.

[6] G. Klein and D. Murray. "Simulating Low-Cost Cameras for Augmented Reality Compositing," IEEE Trans. Visualization and Computer Graphics, vol. 16, pp. 369-380, March 2010.

[7] S.K. Sanjay, G. Adhikari and BK. Das. A Fast Template Matching Algorithm for Aerial Object Tracking: Proceedings of 2011 International Conference on Image Information Processing, 2011[C]. USA: IEEE Computer Society, 2011:978-983.

[8] S. Baker and I. Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework," International Journal of Computer Vision, vol. 56, pp. 221-255, March 2004.