A Motion Vector Visualization Method on Virtual Reality
A Pilot Project in Civic center complex, Sakaiminato

○ Tomohiro Fukuda[*1], Hideki Nada[*2], Hiroyuki Fujii[*3] and Yoann Pencreach[*4]

*1 Associate Professor, Division of Sustainable Energy and Environmental Engineering, Graduate School of Engineering, Osaka University, Ph.D.

*2 Construction manager, Sakaiminato Municipal Office, Ph.D.

*3,4 {Deputy Manager | Development Senior Manager}, Forum8 Co., Ltd.

Keywords: Virtual reality; rendering; shader; virtual reality sickness; motion vector; computer-aided architectural design.

1. Introduction

In recent years, inexpensive head-mounted displays (HMDs) have become commercially available, allowing general users to easily experience virtual reality (VR) content. Also, the creation of VR content by using building information modelling (BIM) and 3D CAD software has become increasingly popular in the fields of computer-aided architectural design (CAAD) and architecture, engineering, and construction (AEC) owing to lower prices and improved production environments.

However, the increased use of VR has resulted in a higher incidence of VR sickness [1]. A likely cause of VR sickness is the mismatch between a user's vision and vestibular senses [2]. During a VR session, there is no vestibular input because the user's body does not move in real space, while the field of view moves according to movement within the VR space. Also, when a designer or VR creator presents along a predetermined fly-through or walk-through path, the audience is forced to be presented with VR images that involve movement and rotation.

VR sickness develops because the visual and vestibular inputs do not match. Lo et al. [3) 4)] found that the severity of VR sickness varies depending on the velocity and direction of the movement within the VR space. Guidelines have been developed to prevent and reduce VR sickness [5)], but the severity of VR sickness is subjective, and the camera settings within a VR application largely depend on the experience of its creators. To ensure the quality of VR content and prevent VR sickness, it is necessary to have a function that can visualise movement within VR space during the VR production process.

We have developed a camera velocity rendering method using a customised segmentation rendering technique that calculates the absolute linear and angular velocities of the virtual camera within a VR space at each frame and overlays a color on the screen according to the velocity value [6)]. This method visualizes the amount of camera movement (XYZ, roll-pitch-yaw) between two frames. As challenges, a single object, and objects that do not move, will have the same value. Also, the value of XYZ is the same even if the depth value from the camera to the object is different. In the real world, objects close to the camera move relatively faster. Therefore, it might be insufficient to evaluate VR sickness.

The objective of this research is to develop a rendering method for a VR platform to meet the challenges of VR. We develop a camera motion vector visualization method using a customised segmentation rendering technique that calculates relative velocities on a VR screen at each frame and overlays a color on the screen according to the value. Our developed rendering method enables approaches that are more versatile than shaded and textured rendering, which is traditionally used in VR. This feature will enable creators to identify where in a virtual scene VR sickness is likely to occur.

2. Literature Review

Concerns about VR sickness are increasing along with the commercialisation and increasing performance of VR systems [1)]. VR sickness occurs while experiencing VR content and has symptoms similar to general motion sicknesses, such as vomiting, coldness and numbness in limbs, nausea, sweating, and headache [7) 8)]. VR camerawork, in particular, has a large impact on VR sickness [3) 4)]. In a prior study, VR sickness was investigated by having subjects complete the Simulator Sickness Questionnaire [9)] as a psychological evaluation and measuring heart rate variability as a physiological evaluation. However, measurement and visualisation of the virtual camera within the VR space are not common in practical VR production processes. Research on reducing VR sickness by galvanic vestibular stimulation is ongoing [10)].

While the severity of VR sickness depends on the individual, collecting data of the VR camera and the degree of VR sickness during user tests is expected to provide useful information through statistical processing. In the previous study, we aimed to

develop functions to measure and visualise the virtual camera in VR content, including linear and angular velocities [6].

3. Visualization of Relative Velocity on the Screen

3.1. CALCULATION

The relative velocity on the screen is the frame of reference (the image that corresponds to a certain moment in the video) and the vector that represents the motion from the frame of reference. It is also known as a motion vector. The value corresponds to the viewing angle velocity (yaw, pitch). When the depth from the camera to an object differs, objects far away from the camera do not move so much and objects near the camera move greatly. Therefore, it can be used as an evaluation factor of VR sickness.

The projection coordinates of the drawing frame are calculated by the formula (1):

$$(x,\ y,\ z,\ w) = M_{Proj} \times MV_{curr} \times Pose_{curr} \times Verte \quad (1)$$

The projection coordinates of the previous frame are calculated by the formula (2):

$$(x',\ y',\ z',\ w') = M_{Proj} \times MV_{prev} \times Pose_{prev} \times Vertex \quad (2)$$

where MV_{curr} is the model view matrix of the drawing frame (inverse of the camera matrix), $Pose_{curr}$ is the object posture matrix for the drawing frame, MV_{prev} is the model view matrix in the previous frame (inverse of the camera matrix) and $Pose_{prev}$ is the object posture matrix for the previous frame.

The inverse matrix of the camera posture of the previous frame and the object pose matrix is maintained and sent to the shader at drawing time.

Next, the relative velocity on the screen is calculated. The screen coordinates of the drawing frame are calculated by the formula (3):

$$(s,t) = \left(\frac{x}{w}, \frac{y}{w}\right), depth = \frac{z}{w} \quad (3)$$

The screen coordinates of the previous frame are calculated by the formula (4):

$$(s',\ t') = \left(\frac{x'}{w'}, \frac{y'}{w'}\right), depth' = \frac{z'}{w'} \quad (4)$$

Therefore, the relative velocity on the screen is calculated by the formula (5):

$$v_s = \frac{s - s'}{\Delta t},\ v_t = \frac{t - t'}{\Delta t} \quad (5)$$

where $\frac{1}{\Delta t}$ is equivalent to the frame rate. The results of this calculation are sent to the shader.

3.2. RENDERING

The VR software UC-win/Road, which has a customisable rendering function, is selected as the VR development platform in this research. Although other VR platforms also have such customisable functions and can be similarly used for development, these platforms can generally execute rendering processing on a screen only once, while UC-win/Road can execute processing multiple times. It can also implement pre-processing and post-processing rendering, both as a whole or for a particular scene. Furthermore, it can define variable values to be delivered to the shader when rendering objects, which can be used on the shader processing side (Figure 1).

Figure 2 shows the processing flow of a customised segmentation rendering that overlays a color on the screen according to the motion vector values. The Shader Calculation process is processed in the plugin when processing Object Drawing in the Main Unit. In the Shader Calculation process, the position on the screen calculated by the formula (4), the relative velocity on the screen calculated by the formula (5), and then output to the frame buffer as a texture based on the calculated relative velocity.

The direction and intensity of the vectors were defined as hue and saturation, respectively, as the color space in which the relative velocity values were mapped. The hues were red (R, G, B) = (1, 0, 0) for the positive x-axis orientation, cyan (R, G, B) = (0, 1, 1) for the negative orientation, green (R, G, B) = (0.5, 1, 0) for the positive y-axis orientation, and magenta (R, G, B) = (0.5, 0, 1) for the negative orientation, as shown in Figure 3. By blending this color space with the original achromatized drawing, we can visualize the relative velocity on the screen.

The shader program is implemented in the OpenGL Shading Language (GLSL) and can be customised by the user. These methods can be implemented as plug-in software on UC-win/Road and can be switched on within the VR platform (Figure 4).

4. Evaluation

We evaluated the method presented in Section 3 by applying it to a building design project in a VR application. The project is the new Sakaiminato civic center complex which includes a hall, a library, conference rooms, a Japanese-style tatami room, a welfare facility, and a disaster prevention facility.

It is not easy to understand the design of a 7,000 square meter building complex using drawings and perspectives from limited viewpoints. In the case of a building complex, there are multiple managers and operators for different functions, and the users' needs are diverse. The new Sakaiminato civic center complex is also planned to have multiple administrators for different functions. Also, it is planned that citizens will participate in the projects and operations of the complex.

The VR for the new Sakaiminato civic center complex was created to review the design from the points of view of the client, administrators and users, and to discover any problems that might

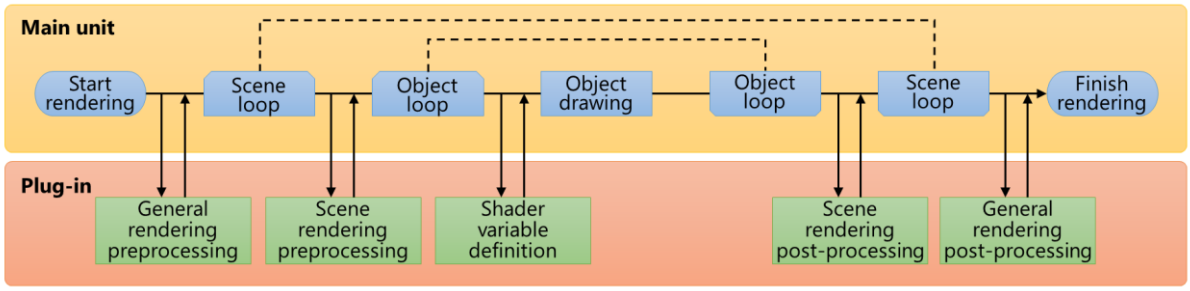

Figure 1. General rendering processing flow.

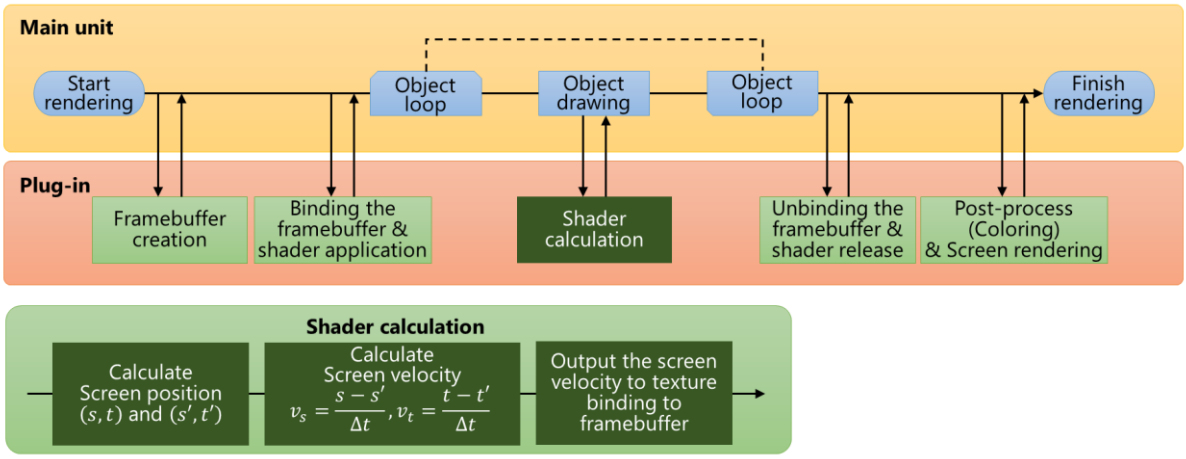

Figure 2. The processing flow of a customised rendering that overlays a colour on the screen according to the motion velocity.

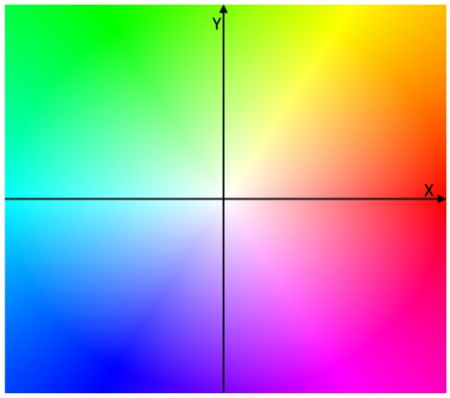

Figure 3. The coordinate system of the overlaid hue, which represents the direction of the motion vector.

Figure 4. Composite of colors representing motion vectors and the original image (monochrome).

appear after the completion of the building, and to study the solutions. The VR will also use publicising the building complex project to the general public.

Architectural VR requires detailed camera work. For example, when turning down a corridor and going up and down the stairs, the speed of the VR virtual camera needs to be carefully monitored to avoid VR sickness. In particular, the entrance space of the library is circular in this project. When the VR walkthrough path in the library moved along a curved path on which furniture such as bookshelves and desks was laid out, the amounts of change in the velocity of the VR camera were observed in a way that would be useful for preventing VR sickness.

We loaded the developed plug-in software and VR content into UC-win/Road, executed the rendering methods according to the usage scene, and observed the results. We used a laptop PC (GALLERIA GCR2070RGF-QC) with Intel Core i7-9700H processor, 16.0 GB of RAM, an NVIDIA GeForce RTX 2070 SUPER 6GB, and a 1920 × 1080 display running Microsoft Windows 10.

The user can switch the rendering methods to the camera's velocity by using the dialogue box in the VR screen. The maximum and minimum values can be set to automatically change the colors on the screen depending on the relative velocity values on the screen. In this case, the maximum value was set to 2.0 and the minimum value was set to 0.0. Typical VR screenshots are shown in Figure 5. The top rows show the normal rendering results and the bottom rows show the motion vector visualization results.

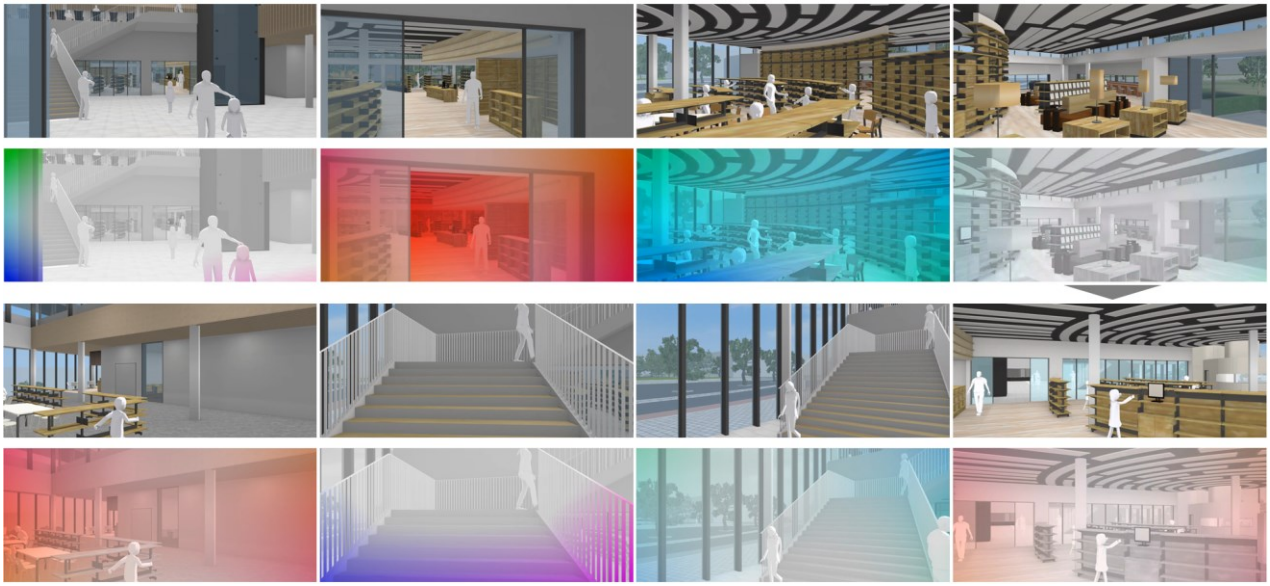

Figure 5. VR screenshots of motion vector visualization in the civic center complex, Sakaiminato (Upper: normal mode, Lower: motion vector visualization).

5. Conclusion

We successfully developed a camera motion vector visualization method using a customised segmentation rendering technique and applied it to a VR application in a design project for a building complex. The developed rendering method could be observed in VR software, and the results were verified.

In the future, the validity of the methods should be evaluated based on actual usage scenarios in architectural design. The maximum and minimum values for the output VR screen were set so that there was a clear color difference for verification of the developed rendering method. Reasonable settings of the maximum and minimum for preventing VR sickness need to be decided through user tests. The occurrence of VR sickness depends on the individual, but collecting data during user tests is expected to provide useful information through statistical processing.

Acknowledgements

The research was partly supported by joint research funding from the Sakaiminato City Office and Osaka University. This research was partly carried out as part of the 2020 research activity of the World16 international research group on virtual reality (Yoshihiro Kobayashi, Kostas Terzidis, Marcos Novak, Marc Aurel Schnabel, Paolo Fiamma, Amar Bennadji, Thomas Tucker, Dongsoo Choi, Matthew Swarts, Ruth Ron, Taro Narahara, Sky Lo Tian Tian, Rebeka Vital and Tomohiro Fukuda). The authors wish to acknowledge the support received from Forum8 Co. Ltd. All images were created by the authors from 2019 to 2020.

References (sample)

1) Kreutzberg, A., 2014, New Virtual Reality for Architectural Investigations, Proceedings of the 32nd eCAADe, 2, 253-260.
2) Dichgans, J., Brandt, T., 1978, Visual-vestibular interaction: Effects on self-motion perception and postural control, Handbook of sensory psychology, Vol.8 Perception, 756-795.
3) Lo, W.T., So, R.H.Y., 1999, Quantifying scene movement with 'spatial velocity' and its effects on cybersickness, Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 43(22), 1219-1222.
4) Lo, W.T., So, R.H.Y., 2001, Cybersickness in the presence of scene rotational movements along different axes, Applied Ergonomics, 32(1), 1-14.
5) Yao, R., Heath, T., Davies, A., Forsyth, T., Mitchell, N., Hoberman, P., Oculus VR Best Practices Guide, http://brianschrank.com/vrgames/resources/OculusBestPractices.pdf, (accessed 2020-09-22).
6) Fukuda, T., Novak, M., Fujii, H., Pencreach, Y., 2020, Virtual reality rendering methods for training deep learning, analysing landscapes, and preventing virtual reality sickness, International Journal of Architectural Computing, Article first published online: Sep. 16.
7) LaViola, J.J.Jr., 2000, A Discussion of Cybersickness in Virtual Environments, ACM SIGCHI Bulletin. 32, 47–56.
8) Groen, E.L. and Bos, J.E., 2008, Simulator sickness depends on frequency of the simulator motion mismatch: An observation, PRESENCE: Virtual and Augmented Reality, 17(6), 584-593.
9) Kennedy, R.S., Lane, N.E., Berbaum, K.S., Lilienthal, M.G., 1993, Simulator Sickness Questionnaire (SSQ): a new method for quantifying simulator sickness, International Journal of Aviation Psychology, 3(3), 203-220.
10) Cevette, M.J., Stepanek, J., Cocco, D., Galea, A.M., Pradhan, G.N., Wagner, L.S., Oakley, S.R., Smith, B.E., Zapala, D.A., Brookler, K.H., 2012, Oculo-vestibular recoupling using galvanic vestibular stimulation to mitigate simulator sickness, Aviation, Space, and Environmental Medicine, 83(6), 549-555(7).