Exploration of Using Face Tracking to Reduce GPU Rendering on Current and Future Auto-Stereoscopic Displays

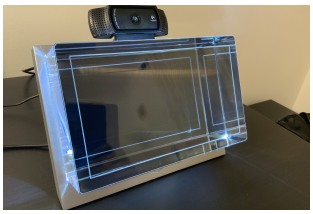
Xingyu Pan
University College Dublin
Dublin, Ireland
xingyu.pan@ucdconnect.ie

Mengya Zheng
University College Dublin
Dublin, Ireland
mengya.zheng@ucdconnect.ie

Abraham Campbell
University College Dublin
Dublin, Ireland
abey.campbell@ucd.ie

(a) Snapshot of 45 views

(b) Snapshot of 6 views



(c) Looking Glass with web camera

Figure 1: The Looking Glass device and run-time snapshots.

ABSTRACT

Future auto-stereoscopic displays offer us an amazing possibility of virtual reality without the need for head mounted displays. Since fundamentally though we only need to generate viewpoints for known observers, the classical approach to render all views at once is wasteful in terms of GPU resources and limits the scale of an auto-stereoscopic display. We present a technique that reduces GPU consumption when using an auto-stereoscopic displays by giving the display a context awareness of its observers. The technique was first applied to the Looking Glass device on the Unity3D platform. Rather than rendering 45 different views at the same time, for each observer, the framework only requires six views that are visible to both eyes based on the tracked eye positions. Given the current specifications of this device, the framework helps save 73% GPU consumption for Looking Glass if it was to render a $8K \times 8K$ resolution scene, and the saved GPU consumption increases as the resolution increases. This technique can be applied to reduce future GPU requirements for auto-stereoscopic displays in the future.

CCS CONCEPTS

• **Computing methodologies** → **Mixed / augmented reality**; • **Human-centered computing** → *Mixed / augmented reality*; Displays and imagers.

ACM Reference Format:
Xingyu Pan, Mengya Zheng, and Abraham Campbell. 2019. Exploration of Using Face Tracking to Reduce GPU Rendering on Current and Future Auto-Stereoscopic Displays. In *Proceedings of SIGGRAPH '19 Posters.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3306214.3338577

SIGGRAPH '19 Posters, July 28 - August 01, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6314-3/19/07.
https://doi.org/10.1145/3306214.3338577

1 INTRODUCTION

The Looking Glass[1] is an emerging Virtual Reality(VR) display technique that displays 45 different views of a virtual object from different angles; each view displays one projection of the object at the respective angle. In this way, user's left eye and right eye can see different views, so as to create binocular parallax. To provide 45 projection views simultaneously, the LCD screen of the Looking Glass renders 45 viewpoints per frame (as in Figure1a), generating a heavy load on the GPU for even the most simplistic of 3D scenes. However, only two views are visible for each observer per frame. This means, when n users are observing the virtual display in front of the Looking Glass, it renders (45-2*n) unnecessary views, which also causes a significant waste of the GPU resource. This waste will be compounded as the screen size and resolution increase in future auto-stereoscopic displays. Accordingly, more than a half of the GPU resources are wasted on these invisible views when there are no more than 10 observers. To avoid this waste, a Unity3D framework was developed that only renders visible views to observers based on their eye positions (shown in Figure1b). This idea originates from Foveated Rendering techniques in virtual reality[Patney et al. 2016] that render the focused area at high resolution while blurring other areas to save the GPU resource. Utilizing this idea, the framework brings context awareness to the original Looking Glass display by enabling the display to adjust itself to the number of observers watching it.

2 IMPLEMENTATION AND EVALUATION

The implementation of the framework can be separated into the following steps:

- Obtain the observers'eye position relative to the device.
- Calculate the visible views to the observers and stop rendering unnecessary views.

[1] https://lookingglassfactory.com/

2.1 Eye position and View Calculation

The relative position of observer's eyes to device can be calculated based on the viewing angle (θ in Figure 2) and vertical distance between the eyes and the device(d_r in Figure 2). To obtain these two figures, a Logitech web camera with a 70.42° Field-of-View (FOV) was fixed on the top center of the device 1c, which captures real-time images to obtain contextual data (user's faces). Next, the DlibFaceLandmarkDetector plugin [2] is applied to analyze the images captured by the camera .

Inter pupil distance(IPD) can be inferred from the distance between the two eyes de_i on captured images. For this current implementation we nearly correct assumption that the IPD will not vary too much, therefore we can generate a linear relationship between d_r and de_i, the equation1 between d_r and de_i is trained through linear regression based on 30 pairs of manually measured d_r and de_i:

$$d_r = -1.25de_i + 106.46 \tag{1}$$

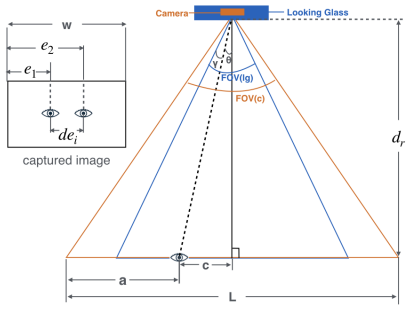


Figure 2: Schematic diagram of the algorithm

With known vertical distance between the user and device d_r, the viewing angle (θ in Figure 2) between the line of sight and the device's normal vector can be get through arc tangent calculation within the triangle formed by the line of sight and the device's normal vector:

$$\theta = \arctan(\frac{c}{d_r}) \tag{2}$$

In the Figure 2, the two isosceles triangles represent the FOV of camera and the Looking Glass within distance d_r, where L is the width of the plane captured by camera in distance d_r. a is the distance between the eye and the left edge of the captured plane. Accordingly, we can get: 3

$$c = \frac{1}{2}L - a \tag{3}$$

In equation 3, the L could be calculated based on the camera FOV (FOV_c) and the vertical distance d_r (4):

$$L = 2 \times \tan(\frac{FOV_c}{2}) \tag{4}$$

The image is a projection of the capture plane, so the ratio between a and L equals to the ratio between e_x and the image width(w):

$$\frac{a}{L} = \frac{e_x}{w} \Rightarrow a = L \times \frac{e_x}{w} \tag{5}$$

Finally, according to 5, 4, 3, and 2, θ can be represented as follows:

$$\theta = \arctan(\frac{\tan(\frac{FOV_c}{2}) \times (1 - \frac{2e_x}{w})}{d_r}) \tag{6}$$

[2]https://github.com/EnoxSoftware/DlibFaceLandmarkDetector

To work out which one of the 45 views is necessary to be rendered for one eye, the angle γ (2) between the line of sight and the left edge of the device's field of view needs to be calculated. According to the Figure 2, γ can be easily represented with the Looking Glass's FOV FOV_lg and θ:

$$\gamma = \frac{FOV_lg}{2} - \theta \tag{7}$$

Among these 45 views, the angle between each two adjacent views is $(\frac{50}{45})°$. Therefore, the index of view k to be rendered is:

$$k \times \frac{50}{45} < \gamma < (k+1) \times \frac{50}{45} \Rightarrow k = (\gamma \times \frac{45}{50}) mod 1 \tag{8}$$

2.2 Performance Evaluation

The effectiveness was evaluated by a comparative experiment to compare the device's GPU (1060 GTX Max-Q) consumption under different resolution settings with and without the aid of this framework. The experiment involved two independent variables, the usage of the face-tracking framework (tracked or non-tracked) and total resolution to be rendered ($2K \times 2K$ to $8K \times 8K$). The only dependent variable is the GPU occupation rate, which was collected from the performance dashboard in task manager.

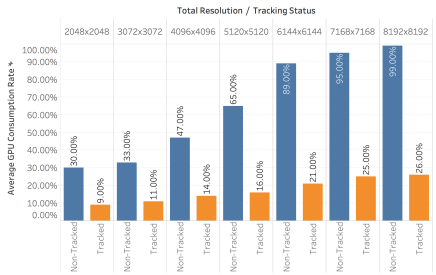


Figure 3: GPU consumption rate of tracked and non-tracked Looking Glass to render 3D objects with different resolution

As Figure 3 shows, the experiment result indicates that the face-tracked device saves 20%-73% GPU consumption compared to the non-tracked device. Moreover, the result also shows that the GPU consumption saved by the face-tracking framework continuously increases from 20% with $2K \times 2K$ resolution to 73% with $8K \times 8K$ resolution. The evidence presented thus far supports the idea that our technique helps save GPU consumption and this saving as a percentage will only increase in the future.

3 CONCLUSIONS AND FUTURE WORK

This technique has been demonstrated to help save the GPU consumption of existing and future auto-stereoscopic display. Despite that the face-tracking framework saves significant GPU consumption for the Looking Glass, there still exists some restrictions on it. To improve the face-tracking framework, we plan to replace the web camera with depth cameras or use machine-learning techniques to detect the eye-device distance with a higher accuracy.

REFERENCES

Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 179.