Gaze Tracking Optimization for Virtual Reality

Timothy Southwick
University of Massachusetts Lowell
Timothy_Southwick@student.uml.edu

Abstract

Gaze tracking in virtual reality headsets could be used to simulate depth of field, simulate high dynamic range, and more. To get high precision and sample rate, certain algorithms and hardware may prove useful.

1. Introduction

Virtual reality headsets are intended to visually immerse their users in a 3D space. To do this, they use motion tracking, high resolution displays operating at high frame rates, and a lot of processing power for both CPU and GPU. Consumer models are out of the budget of most potential casual users, and while very powerful, certain sacrifices had to be made to keep the headsets affordable for their initial launch to high-end consumers. Among those sacrifices is gaze tracking.

While head tracking may be enough to determine what to draw on the display, gaze tracking can significantly improve a VR experience. Improvements include adjusting the brightness of the images displayed to match pupil dilation; applying a depth of field effect to match eye gaze; determining how much processing power is needed in different regions of the display; and logging a user's gaze during the development and testing of a VR experience.

However, high end processing power is required to use VR, with or without gaze tracking. The resolution and frame rate are higher than a typical monitor. Any gaze tracking solution needs to match the frame rate of the display, have a sufficiently high resolution that the simulated gaze is within a few pixels of their real gaze, be capable of running in real time, and the optics required for it to function must fit within the VR headset.

2. Camera hardware requirements

While gaze tracking is possible for visible light cameras, reflections in the eye make this less than ideal. However, using near infrared light makes the pupil stand out for gaze tracking algorithms. This should not require a cost difference, since typical digital cameras are sensitive to near infrared light; all that is needed is a filter in front of the lens to block visible light.

The form factor of the camera must be small enough to fit in the headset, while still providing sufficient resolution and frame rate. The cameras in smart phones meet that requirement.

A single camera could be used, however it would require an extra step of processing to identify where the eyes are in the frame before locating the pupils within the eyes. Additionally, this limits the resolution dedicated to each eye, which should be maximized to be nearly the resolution of the display. Using two cameras would be preferable, although it would require additional bandwidth.

The bandwidth of the cameras doesn't need to be sent from the headset to a computer – only four angles need to be sent, two per eye, as well as the relative size of each pupil compared to either the eye or iris. If the processing power required is small enough, this can be done within the headset not too far from the gaze tracking camera(s).

3. Random access cameras

There is potential for an optimization in the camera, in that after the eye is found in one frame, only part of the following frames is needed until the eye is lost, at which point a full frame may be required again. By obtaining smaller sections of the sensor data as needed, the amount of data sent from the sensor to the processor can be reduced for each frame, which could allow for higher frame rates.

Two options are available for randomly accessing an image through camera hardware: optically and electrically.

Through a series of lenses, mirrors, and liquid crystal panels, it is possible to have a single sensor view through multiple lenses at different angles, producing sections of a larger image [1]. This may be repeated in multiple stages to allow access to any section similar to a binary tree. This means that the optical sensor can have a lower resolution, which

allows for higher frame rates. However, it also adds to the weight of the headset, reduces the amount of light received by the sensor, and potentially distorts the image.

To electrically provide random access to the image sensor, the data paths need to be modified such that there are essentially multiple sensors placed directly next to each other. Having these extra data paths may require adjusting the space between the pixels, produce noise from cross-talk between neighboring data lines, and if the sensor has to be custom made, significantly increase development cost. Alternatively, this can be done with per-subpixel access instead of block access [2].

However, either method would allow for significantly higher frame rates, as only the edges of the pupil need to be tracked once they have been located. For most frames the eye will have barely moved from where it was, and if the eye moves too fast a full frame can always be taken as needed.

4. Existing gaze tracking algorithms

There are many existing gaze tracking algorithms available today. Fortunately, in this case we can ignore the tracking of where the eyes are in an image, since the eyes are at a known location relative to the eyes. Tracking the location of the eyes within the frame can still be performed to compensate for a loose headset if desired. If an eye must be localized for the first frame, and there is no information for where the eye could be to limit the pixels to search, it is possible to reduce the resolution for faster data access and still get reasonably accurate results [3].

There are multiple methods of determining the location of a pupil within an eye once the eye is found. The circular Hough transform assumes that the pupil is a circle, and that all points on the edge of a circle are equidistant to the center of the pupil, which has a single radius [4]; this works well until the pupil is viewed from too far to the side and appears to be an ellipse, and modifications to handle cases like this increase the complexity of the algorithm rather fast. The Starburst algorithm is an iterative approach that starts tests how far a given point is from the edges along several rays, and uses the point as the center when the desired accuracy is achieved [5]. It handles non-circular pupils very well, but requires several iterations if it starts searching from the wrong location. It should be able to run with far fewer iterations if it starts searching inside the pupil, especially if starting near the center of the pupil.

I like to address the growing trend of using machine learning to handle problems. While machine learning may be able to get the most accurate results, it is not appropriate where speed is important. The minimum frames per second for a human to interpret motion instead of still images is 24 frames per second, television runs at 25 (PAL) or 30 (NTSC) frames per second, computer monitors run at 60 frames per second, and typical virtual reality headsets for computers require a bare minimum of 90 frames per second on average. If a machine learning algorithm runs at only 15 frames per second, it is far too slow to be used in virtual reality [6].

5. Gaze tracking algorithm improvements

Currently, gaze tracking solutions need to get a frame, remove noise, identify an remove reflections that may interfere, and from there they differ. Typically some form of edge detection is performed, points are found on the edge of the pupil, and the center of the pupil is extrapolated. Every step between removing noise and edge detection loops over every pixel in the frame, and for each pixel creates a pixel in a new image buffer by looping over pixels in a square around the selected pixel.

$O(n^4)$ sounds like a very inefficient algorithm; a GPU helps significantly with this, being able to handle multiple pixels at once with the same command, but the resolution for gaze tracking will need to be nearly as large a resolution as the images being rendered for the display. However, since the eye is unlikely to move far in any given frame, the frame only needs to be processed within a few degrees of the pupil's last location.

Additionally, since the position in the current frame is nearly identical to that in the previous frame, the edges of the pupil can be found with linear edge detection on either side of the previous edge at predetermined points – for instance, every 1/16 of a circle, checked radially for a range of radii between 90% and 110% of the last location. Linear edge detection is done using only the difference in brightness from the current and last pixel checked, which is an O(n) algorithm. A minor speed improvement from this is to do linear edge detection running either vertically or horizontally instead of diagonally.

6. Gaze tracking applications for virtual reality

Gaze tracking can significantly improve a VR experience. Pupil dilation can be used to adjust the brightness of a scene. By finding the intersection of the gaze of both eyes, the point the user is focusing on can be determined; I shall call this a fixated point. Fixated points can be used to log where a user has been looking for development, testing, research, and inspection training [7]. Fixated points can also be used as an input to a depth of field effect in a scene, which

may alleviate eye strain or be used for aesthetic purposes, especially in combination with a display with an infinite depth of focus. For one implementation of such a display, I would suggest looking into Steve Mann's aremac, which I feel does not get the attention it deserves. While I have not been able to find any documentation about the device by itself, Mann uses it in several head mounted devices, and as I understand it, it acts as a counterpart to a pinhole camera.

Knowing the fixated point of the user doesn't just offer features that require more processing power; it also allows for lowering the quality of areas the user is not paying attention to. To stop rendering those areas entirely might be noticed out of the corner of the eye, but they can be rendered at a lower resolution, and with less detail.

Additionally, rapid motion of the fixated point means the user is looking suddenly in another direction. During that motion, the user's eyes and/or brain are unable to process what is in front of them as well as if they turned their head; this may provide another opportunity to lower quality briefly, but testing would be required to ensure this is the case.

7. Conclusion

Gaze tracking in VR headsets can be an expensive feature to add, but time-based optimization can make it feasible at high resolutions and refresh rates without requiring the processing power of obtaining the first frame. The bandwidth required may be large, but by requesting only what is needed, it is possible to get more meaningful data while still reducing the current bandwidth requirements. The benefits gaze tracking provides can improve immersion, improve testing during development and research, and reduce unneeded processing time.

8. References

[1] N. C. Gallagher and J. P. Allebach and P. Haugen and D. Kranz, "Random access camera", *[1991] Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems Computers*, IEEE, 1991.

[2] Suraj Bhaskaran and Herb Ziegler and Claudia Borman and John Swab and Carey Beam and Tony Chapman and John Capogreco and Steve VanGorden and Mark Greco and Matt Pace and Zulfiqar Alam and Jon Miles and Mike Pilon and Joe Carbone and Helen Jung and Rindy Finney and Linda Lee and Victor Tsai and and Arnold London and Bruce Meyers, "SPECTRACAM TM – Random Access Charge Injection Device Cameras for Spectroscopy", *Onieda Research Services, Inc.*, Onieda Research Services, Inc., 2004.

[3] George, Anjith and Routray, Aurobinda, "Fast and Accurate Algorithm for Eye Localization for Gaze Tracking in Low Resolution Images", *IET Computer Vision*, IET, 2016.

[4] R. G. Bozomitu and A. Păsărică and V. Cehan and C. Rotariu and C. Barabaşa, "Pupil centre coordinates detection using the circular Hough transform technique", *2015 38th International Spring Seminar on Electronics Technology*, ISSE, May 2015.

[5] Li, Dongheng and Winfield, David and Parkhurst, Derrick J, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches", *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, IEEE, 2005.

[6] Kyle Krafka and Aditya Khosla and Petr Kellnhofer and Harini Kannan and Suchendra Bhandarkar and Wojciech Matusik and Antonio Torralba, "Eye Tracking for Everyone", *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.

[7] Duchowski, Andrew T. and Shivashankaraiah, Vinay and Rawls, Tim and Gramopadhye, Anand K. and Melloy, Brian J. and Kanki, Barbara, "Binocular Eye Tracking in Virtual Reality for Inspection Training", *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, ACM, Palm Beach Gardens, Florida, USA, 2000