Latency Compensation by Horizontal Scanline Selection for Head-Mounted Displays

Jason Jerald*     Andrew Fuller     Anselmo Lastra     Mary Whitton     Luv Kohli     Fred Brooks
University of North Carolina at Chapel Hill

ABSTRACT
A fundamental task of a virtual-environment system is to present images that change appropriately as the user’s head moves. Latency produces registration error causing the scene to appear spatially unstable. To improve the spatial stability of the scene, we built a system that, immediately before scanout to a head-mounted raster display, selects a portion of each scanline from an image rendered at a wider display width. The pixel selection corrects for yaw head rotations and effectively reduces latency for yaw to the millisecond range. In informal evaluations, users consistently judged visual scenes more stable and reported no additional visual artifacts with horizontal scanline selection than the same system without. Scanline-selection hardware can be added to existing virtual-reality systems as an external device between the graphics card and the raster display.

Keywords: Virtual environments, latency, real-time graphics, hardware, head-mounted displays, system architecture, spatial stability, human factors

1. INTRODUCTION
A virtual-environment (VE) system presents visual scenes that change appropriately as a user moves. System delay is defined as the true end-to-end delay of the entire system from the time of reading head pose to the time of displaying a pixel computed using that pose. In this paper, we define latency as the effective delay of the system as perceived by the user. Without prediction or correction, latency is equal to system delay. Registration error is the difference between where a pixel appears to be and where the pixel would appear if it were a physical object in the real world. These errors can further be classified as static or dynamic. Static errors result from optical distortion, incorrect viewing parameters, imprecise calibration of equipment, etc., and occur even when the user keeps the head completely still. Dynamic errors result from temporal mismatch between head movements and the visual display.

Latency causes dynamic errors when there is head movement. As latency and head velocity each increase, dynamic errors increase. This causes the virtual environment to appear to be spatially unstable as users move their heads. Experiments show observers detect spatially unstable scenes that amount to +/- 3% of head movement.

Due to the coupling of the display with the user’s head, latency causes more dynamic registration error with Head-Mounted Displays (HMDs) than with world-fixed displays (e.g., CAVE®s). For a world-fixed display, latency causes no error for virtual objects located at the display surface and error increases with object distance from the display surface. For an HMD, when the user rotates to the left the entire scene must move to the right for the virtual world to appear spatially stable. Even for moderate head velocities, typical latencies cause more registration error than all other registration errors combined. Pixel error due to orientation is independent of object distance whereas pixel error due to translation decreases with object distance. Head rotations cause more pixel error in HMDs than head translations for all but the closest objects. For a system with 30 ms of latency and a typical head movement at 50 degrees per second and 0.5 meters per second, error due to orientation is 1.5 degrees and error due to translation is 0.85937 degrees for objects at 1 meter and 0.085943 degrees for objects at 10 meters.

In this paper, we describe a method to compensate for delay-induced yaw (left/right head rotation) error. The system renders the scene with a wider than necessary field of view, and then, at the last moment before scanning a line out to the HMD, selects the appropriate portion of the scanline. For the purposes of this paper, we assume the HMD is a raster display.

*jjerald@cs.unc.edu; http://www.cs.unc.edu; University of North Carolina, CB 3175, Sitterson Hall, Chapel Hill, NC, 27599-3175
2. DISPLAY DELAY

Much of the research on system delay has focused on delay measured up to the start of a frame in the video output signal, ignoring the delay of the display technology. In this section, we focus on the latter.

Liquid crystal displays (LCDs), commonly found in modern HMDs, often have more delay than the sum of all other components of the system. Table 1 shows average response times we measured with our lab’s Virtual Research V8 and Sony Glasstron HMDs. These numbers were the smallest we obtained; larger response times were measured under other conditions. The response time of LCDs also varies depending on starting and ending intensities and does not follow a linear relationship.

Fast response displays are necessary when precise dynamic registration is required. CRTs have a phosphor-response time on the order of microseconds. Other fast response microdisplay technologies do exist, but care must be taken in choosing such displays, as they often introduce additional delays due to internal buffering (up to a full-frame time).

Studies suggest that with HMDs humans can perceive latency of less than a typical frame (16.7 ms at 60 Hz). Since, for a raster display, the bottom pixels of a display appear later than the top pixels, we must focus upon intra-frame delay to reduce latency below the perceptual threshold of humans. A standard frame renderer renders the entire frame from a single precise point of view. If the viewpoint is in fact moving, only a single pixel is correct. An accurate system would render every pixel from a new viewpoint.

<table>
<thead>
<tr>
<th>display response (ms)</th>
<th>16.7%</th>
<th>50%</th>
<th>83.3%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Research V8 HMD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>black to white</td>
<td>16.5</td>
<td>31.6</td>
<td>73.3</td>
</tr>
<tr>
<td>white to black</td>
<td>14.5</td>
<td>33.9</td>
<td>113.3</td>
</tr>
<tr>
<td>Sony Glasstron HMD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>black to white</td>
<td>18.8</td>
<td>40.6</td>
<td>101.7</td>
</tr>
<tr>
<td>white to black</td>
<td>19.8</td>
<td>43.6</td>
<td>123.2</td>
</tr>
</tbody>
</table>

Table 1. Measured pixel response times of the LCDs in our V8 and Glasstron HMDs (ms). These response times are a large portion of total system latency. Total system latency varies greatly depending on the percentage of intended intensity that is desired.

3. LATENCY COMPENSATION

Because computation is not instantaneous, virtual-environment systems will always have delay. Two techniques, prediction and post-rendering manipulation, can mitigate delay-induced latency.

Head-motion prediction is a commonly used latency compensation technique for VR systems. This produces reasonable results for small delays or slow head movements. However, increased delays increase motion overshoot and amplify sensor noise. Furthermore, prediction is incapable of instantaneous response to rapid transients.

Post-rendering techniques first render an image larger than the final display and then, late in the display process, select the appropriate subset to be presented to the user. Environment mapping allows quick viewing of complex worlds by projecting the world onto the six sides of a large cube or a cylindrical panorama. For objects at a reasonable distance from the user, rotations can be approximated by translations with little registration error. Head rotation simply alters what part of display memory is accessed—no other computation is required. Environment maps do not correct for motion parallax and thus are not appropriate for geometry close to the viewpoint.

Regan and Pose took environment mapping further by projecting geometry onto concentric cubes surrounding the viewpoint. Larger cubes that contain projected geometry far from the viewpoint do not require re-rendering as often as
smaller cubes that are close to the viewpoint. In order to minimize dynamic registration error due to large translations, a full 3D warp\textsuperscript{10} or a pre-computed light field\textsuperscript{11} is required.

Environment mapping can be simplified by projecting the world onto a single image plane. The image plane is then shifted appropriately to reduce yaw and pitch error\textsuperscript{12-19}. This technique works well if the user is looking in the same general direction as the original projected image plane.

4. SCANLINE SELECTION

We go beyond shifting entire images by selecting portions of scanlines from an earlier rendered image. The selection is determined by the yaw-angle offset (the difference of the rendered orientation and the current orientation just before scanout). The system measures the yaw-angle offset $\theta$ by integrating the voltage output from a gyroscope over the time of the system delay.

Figure 1 shows high-level diagrams of a typical VE system and the same system with scanline-selection hardware added. Figure 2 provides a high-level algorithm describing operation of a scanline-selection system.

Scanline-selection hardware takes output from the rendering computer and outputs video with corrected images to an HMD. No other communication is required between the rendering computer and the scanline-selection hardware; the hardware is plugged into existing systems with no modifications to the original application on the rendering computer.

---

**Figure 1.** High-level block diagram of a typical VE system and the same system with scanline selection

---

**Figure 2.** System algorithm

For every frame:

Get the most recent tracker update
Render a larger than final output frame from a single viewpoint for time $t_0$
Start scanning out the video signal from the rendering computer to the scanline-selection hardware
For every scanline, $n$, in a frame, the FPGA does the following:
Integrate a signal from a gyro (equivalent to angular velocity) from time $t_0$ to the current scanline time $t_n$
Approximate the error of the angular displacement by projecting the angle onto the viewing plane
Compensate for yaw error by selecting the current scanline from the larger rendered image
Scan out the current scanline to the HMD
4.1 Calculation of scanline selection

The linear offset \( s \), the amount the scanline is to be shifted in meters, is determined by projecting the yaw-angle offset \( \theta \) onto the image plane.

\[
s = D \tan \theta
\]

where \( D \) is the distance from the viewpoint to the image plane. \( s_{\text{pixels}} \) is the linear offset in pixels

\[
s_{\text{pixels}} = s \left( \frac{w_{\text{pixels rendered}}}{w_{\text{rendered}}} \right)
\]

where \( w_{\text{rendered}} \) is the width of the rendered image plane in meters and \( w_{\text{rendered pixels}} \) is the width of the rendered image plane in pixels.

The system then selects a portion of the input scanline and sends this portion of the scanline to the HMD. The pixels selected from the rendered scanline to be displayed in the HMD are the pixels in the range

\[
\left( s_{\text{pixels}} + \frac{w_{\text{rendered Pixels}} - w_{\text{displayed Pixels}}}{2}, s_{\text{pixels}} + \frac{w_{\text{rendered Pixels}} + w_{\text{displayed Pixels}}}{2} \right)
\]

where \( w_{\text{displayed Pixels}} \) is the width, in pixels, of the display in the HMD.

4.2 VE system requirements

The following are not absolutely necessary for scanline-selection hardware to reduce latency. However, they are required to achieve optimal results.

- **Fast response raster display.** Most modern CRTs have fast-response phosphor in the microsecond range. Newer microdisplay technology is becoming available, but one must be careful of internal buffers that add additional delay.
- **Consistent system delay.** The scanline-correction hardware must know when to start integrating the gyroscope signal. This start of integration time is most easily determined by having a known consistent delay in the system. Consistent system delay would not be required if an additional signal was sent once per frame from the VR system to the scanline-selection hardware to signal the start of integration time.

4.3 A prototype system

For display, we modified an Eyeegen 3 CRT HMD. This HMD originally displayed buffered color sequential NTSC fields at 30 Hz but was modified to allow our hardware to directly drive the CRT through a monochrome VGA signal with a resolution of 456x234 pixels at 180 Hz. For tracking, we used a 3rdTech Hi-Ball 3000 tracking system (updates at ~1000 Hz and accurate to within 1 mm and 0.02°). For rendering, we used a Dell Precision Workstation 530 equipped with dual 1.69 GHz processors, an NVIDIA GeForce 6600 GT, Windows XP, VRPN [19], and the UNC EVE research team’s Effective Virtual Environment Intermediate Layer (EVEIL).

Figure 3 shows a picture of the hardware with the components labeled. Each component is duplicated to allow left and right eye selection. All boards are custom built other than the Digilab IIE development board that includes a Field Programmable Gate Array (FPGA). Figure 4 shows a block diagram of the logical components for a single eye.

To be compatible with the custom-built hardware and HMD, the graphics processing unit (GPU) on the PC is configured to output 640x234 video at 180 Hz. At the beginning of each frame of video output (the starting edge of the vertical sync signal), the PC grabs the most recent real-time head-pose data from the tracker (mounted with the gyroscope on top of the HMD). The rendered video is transmitted as DVI, which is decoded by a DVI input board and digitally sampled by an FPGA. Scanlines are stored in a single line buffer, which adds one scanline of system delay (~20 µs).
At the beginning of each scanline, a customized analog-to-digital conversion (ADC) board takes a single 12-bit reading from the gyroscope that provides a voltage proportional to angular velocity. The digitized readings are integrated within the FPGA to report the amount of head rotation that has taken place since the start of the current frame. Additional integration registers store previous frame integrations from the time of the Hi-Ball head-pose report used for rendering to the beginning of the current frame. The registers are added together to determine the pixel offset required to compensate for total delay over multiple frame times.

The FPGA outputs video at a screen resolution of 456x234, also at 180 Hz. At the start of each scanline, the FPGA computes the left or right offset from center (determined from the current value of the sum of the integration registers). A 456 pixel scanline window is selected from the 640 pixels rendered so that the pixels correspond with the user’s most-recent head orientation. Figure 5 shows conceptually the large image rendered on the PC, scanline selection with no head movement (zero offset), and scanline selection with right-to-left head movement (increasing offset with increasing system delay). Although this static representation of the dynamic scene appears skewed, the image does in fact appear more stable when moving the head due to the timing difference of the scanlines.

The outgoing digitized VGA signal is converted to an analog grayscale video signal by an eight-bit video digital-to-analog conversion (DAC) board. The video signals are converted into differential signals by a differential driver board. The signals are driven over a 50 foot single-end terminated line and converted back into ground-referenced signals to drive the HMD.
Figure 4. Block diagram of the scanline-selection hardware

Figure 5. Conceptual scanline selection from a rendered image

- Large Image Rendered for pose at time $t_0$
- Scene displayed with no head movement
- Scene displayed with right-to-left head movement
4.4 Error Analysis

Scanline selection is an approximation to a VE system with zero latency. Error is minimal at the center of the display and grows towards the edges of the display. In this section, we describe the error of image shifting as shown by Mazuryk and Gervautz.\(^{15}\)

Let the origin be located at the viewing point for the originally rendered image with the x-axis set in the right-hand direction as shown in Figure 6. Let \(P_0\) be the location of a pixel on the original rendered image, \(P_s\) be \(P_0\) shifted by \(s\), \(P_r\) be \(P_s\) rotated by the yaw-angle offset \(\theta\), and \(P_p\) be \(P_r\) projected onto the original rendered plane. Let \(x_0\) be the x-coordinate of the point \(P_0\) and \(\varphi_0\) be the angle between the center of the rendered image and \(x_0\) so that

\[
\varphi_0 = \tan^{-1}\frac{x_0}{D}
\]  

(4)

and

\[
x_0 = D\tan\varphi_0.
\]  

(5)

The x-coordinate of the point \(P_s\) is

\[
x_s = x_0 + s = -D\tan\varphi_0 + D\tan\theta.
\]  

(6)

The angle between the center of the rendered image and \(x_s\) is

\[
\varphi_s = -\tan^{-1}\frac{x_s}{D} = \tan^{-1}(\tan\varphi_0 - \tan\theta).
\]  

(7)

Rotating \(P_s\) by \(\theta\) gives the angle between the center of the rendered image and \(P_r\):

\[
\varphi_r = \varphi_s + \theta = \tan^{-1}(\tan\varphi_0 - \tan\theta) + \theta.
\]  

(8)

The yaw error at \(P_0\) is the difference between \(\varphi_r\) and \(\varphi_0\):

\[
\Delta\varphi = \varphi_r - \varphi_0 = \tan^{-1}(\tan\varphi_0 - \tan\theta) + \theta - \varphi_0
\]  

(9)

Without scanline selection error is constant across the entire display

\[
\Delta\varphi = \theta
\]  

(10)
Figure 6. Geometric representation of scanline selection as an observer turns his head to the left by \( \theta \) degrees. The system shifts the image to the right by \( s \) to compensate for \( \theta \). The pixel at \( P_0 \) shifts to \( P_s \). The observer rotates his viewpoint, transforming the point \( P_s \) to \( P_r \). \( P_p \) is \( P_r \) projected into the rendered scanline space. The error angle \( \Delta \varphi \) is the angle between the points \( P_0 \), the point where a pixel should theoretically appear, and \( P_p \), the point where the pixel appears after scanline selection.

Figure 7 shows a plot of the error \( \Delta \varphi \) for 100 ms of system delay and a head turn of 100 degrees per second, i.e. a yaw-angle offset \( \theta \) of 10 degrees. Little error occurs in the center of the display. Towards the periphery error approaches the constant 10 degrees of error that would occur without scanline selection.
Figure 7. Horizontal error with scanline selection for latency of 100 ms and head rotation of 100 degrees/second. Pixels in the center of the display have little error and error increases towards the periphery. No scanline correction would result in a constant error of 10 degrees. The curve has a similar shape for other conditions with maximum error not exceeding error with no scanline selection.

5. SYSTEM EFFECTIVENESS

We demonstrated the system to seven male computer science graduate students for a preliminary determination of horizontal scanline-selection effectiveness for stabilizing scenes. Participants were able to toggle between scanline selection and no scanline selection via a handheld button. They were not told the details of the manipulation.

Although our VE system without scanline correction is capable of approximately 11 ms of system delay (from the time of head movement to the top-most scanline being displayed), we inserted various additional delays (totaling 89, 44, 22, and 11 ms) along with the appropriate scanline selections, so that participants judged scanline-selection effectiveness under different system delays. Participants were encouraged to make fast yaw movements as well as circular movements that included both yaw and pitch.

All participants preferred scanline selection over no scanline selection for 89 and 44 ms of system delay. All but one preferred scanline selection for 22 ms of system delay. Three of the seven preferred scanline selection for 11 ms of system delay. Though some participants gave no preference in the low system delay conditions (those conditions in which latency was already difficult to detect), none of the participants preferred no scanline selection over scanline selection for any condition.

6. FUTURE WORK

We plan to perform a more formal user study to determine the effectiveness of scanline selection.

The system could be modified to reduce pitch error by vertically selecting scanlines. Vertical scanline selection requires additional buffering for storing and retrieving multiple scanlines. This buffering would cause additional system delay as multiple scanlines must be sent to the hardware before selection for scanout to the HMD.

Error could further be reduced by first predicting head position, then performing scanline selection.
We are also in the process of building a true just-in-time scanline-rendering system to study latency perception. This system will not contain the error described in section 4.4. A true scanline-rendering system would also remove error due to roll and translations of the head. However, due to computational requirements of rendering every scanline (at 46.98 KHz in our system), the scenes will be limited to simple geometry. For complex scenes, we believe scanline selection to be a better solution for reducing latency.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge hardware support, discussion, and feedback from John Thomas, David Harrison, Bernard Adelstein, Steve Ellis, Henry Fuchs, Chris Oates, Eric Burns, and Rick Jerald. This research was facilitated in part by a LINK Fellowship, a North Carolina Space Grant Fellowship, a National Physical Science Consortium Fellowship, and by stipend support from HRL Laboratories. Partial support was provided by grants CCF-0306478 and CCF-0205425 from the National Science Foundation. This work used equipment purchased through National Institutes of Health NIBIB EB002025.

REFERENCES