Construction and Evaluation of an Ultra Low Latency Frameless Renderer for VR

Sebastian Friston, *Student Member, IEEE*, Anthony Steed, *Member, IEEE*, Simon Tilbury and Georgi Gaydadjiev, *Member, IEEE*

Abstract—Latency - the delay between a user's action and the response to this action - is known to be detrimental to virtual reality. Latency is typically considered to be a discrete value characterising a delay, constant in time and space - but this characterisation is incomplete. Latency changes across the display during scan-out, and how it does so is dependent on the rendering approach used. In this study, we present an ultra-low latency real-time ray-casting renderer for virtual reality, implemented on an FPGA. Our renderer has a latency of ~1 ms from 'tracker to pixel'. Its frameless nature means that the region of the display with the lowest latency immediately follows the scan-beam. This is in contrast to frame-based systems such as those using typical GPUs, for which the latency increases as scan-out proceeds. Using a series of high and low speed videos of our system in use, we confirm its latency of ~1 ms. We examine how the renderer performs when driving a traditional sequential scan-out display on a readily available HMD, the Oculus Rift DK2. We contrast this with an equivalent apparatus built using a GPU. Using captured human head motion and a set of image quality measures, we assess the ability of these systems to faithfully recreate the stimuli of an ideal virtual reality system - one with a zero latency tracker, renderer and display running at 1 kHz. Finally, we examine the results of these quality measures, and how each rendering approach is affected by velocity of movement and display persistence. We find that our system, with a lower average latency, can more faithfully draw what the ideal virtual reality system would. Further, we find that with low display persistence, the sensitivity to velocity of both systems is lowered, but that it is much lower for ours.

Index Terms—Low Latency, Frameless Rendering, Image Quality, Ray Casting, Hardware Acceleration

1 Introduction

Frameless rendering is a method that involves updating pixels on a display in an arbitrary, application-defined order [2]. This is in contrast to traditional frame-based algorithms, such as the painter's algorithm, where individual frames are rendered and only then displayed to the user in sequence [10]. The requirement to render a complete frame before it can be used places a lower limit on latency.

Latency is commonly defined as the delay between a user's action and the response to this action. It is a result of the processing, transport, sampling and scan-out delays of the components that make up a Virtual Envrionment (VE) [18]. Latency is highly detrimental to Virtual Reality (VR) in terms of both user comfort and performance (e.g. [3, 15]). However, because it is inherent to the current technology used to build VEs, it cannot be entirely eliminated. Frameless renderers have been proposed as a way to minimise the apparent latency of interactive systems [2]. Since they do not need to wait on any single frame, the delay between user input and the response can be significantly reduced. This is most clearly illustrated in examples utilising image warping, where the tracker is separated from the display by only a few image transformations (e.g. [29, 23] - though many systems using a variety of techniques have been proposed).

Authors typically compare the performance of frameless rendering algorithms to traditional algorithms in terms of spatial quality. A number of studies have demonstrated that frameless renderers can match traditional renders for quality. Less time has been spent examining the behaviour of a frameless renderer in terms of latency however. Further, frameless renderers must still be integrated with existing technologies to form a VE system. While frameless rendering ameliorates the requirement of double-buffering to avoid image tearing, most display technologies still use a sequential scan-out. In this study we combine an Oculus Rift DK2 with a custom frameless renderer to form a real VE with which to investigate this.

We present a new ultra-low latency frameless renderer based on hardware accelerated ray-casting on a Field Programmable Gate Array (FPGA). Like Regan & Pose's Address Recalculation Pipeline [25] or Regan et al.'s light field renderer [26], our frameless render works on the principle of "just in time pixels" where the computation for a single pixel is completed only moments before it is transmitted to the display. This facilitates changing the viewing transforms to reflect the latest tracking data during scan-out, and minimising the latency of the rendering stage as far as possible. The use of ray casting means individual pixels can be computed in an arbitrary order and used immediately. By implementing the ray-caster in hardware we reduce the latency of our rendering stage to within a millisecond. Using a combination of high and low speed video captures, and objective Image Quality Measures (IQMs) suitable for Computer Generated (CG) imagery, we investigate the behaviour of this renderer and compare it with that of an equivalent system built using a GPU.

We describe how the latency across a display is not constant, as it is usually considered, but varies as the visible pixels drawn on the display age during the scan-out process. The relative age of different regions of the screen is not the same between rendering techniques. Our frameless render has a latency less than one frame period, and so the lowest latency regions of the display follow the scan-beam. A frame-based GPU system on the other hand begins scanning out an already aged image from the same location on each frame. The implications are not just a lower average latency for the frameless system, but an inherently different composition. The frameless renderer displays an amalgamation of 13 ms worth of tracker history manifested as a skew feature under motion. To assess the consequences of these differences we generate ground truth renders to emulate an ideal VE, with a zero latency tracker, renderer and display, each running at 1 kHz. Using image quality metrics suitable for CG imagery we judge the fidelity of the two renderers and how it varies with display persistence and tracker velocity.

- *Sebastian Friston is with University College London. E-mail: sebastian.friston.12@ucl.ac.uk.*
- *Anthony Steed is with University College London. E-mail: a.steed@cs.ucl.ac.uk.*
- *Simon Tilbury is with Maxeler Technologies Ltd. E-mail: simon@maxeler.com.*
- *Georgi Gaydadjiev is with Maxeler Technologies Ltd. E-mail: georgi@maxeler.com.*

Manuscript received 21 Sept. 2015; accepted 10 Jan. 2016. Date of publication 20 Jan. 2016; date of current version 19 Mar. 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2016.2518079

In Section 2 we review frameless rendering as applied to VR, and contrast existing hardware accelerated renderers with our own. We also provide a brief overview of image quality measures, specifically where applied to CG imagery and for purposes other than compression artefacts (the typical application of such measures). In Section 3 we describe the apparatus used for our investigation. This section includes an in-depth description of the operation of our ultra low latency renderer. In Section 4 we describe the results of our investigation into the two systems. We first discuss how the latency of each system was measured and give the results. We then describe how each system renders to the display and the implications for the latency and the resulting composition seen by the user. Section 4.3 covers in detail the procedure for capturing and analysing real-time systems, justification for our chosen IQMs, and a discussion on how the measured fidelities relate to the behaviour we observed in Section 4.2. Our conclusions are presented in Section 5.

2 Previous Works

2.1 Frameless Rendering in Virtual Reality

Latency cannot be removed given current technology, so a number of authors have designed methods to compensate for it. Regan & Pose created the Address Recalculation Pipeline, which decoupled the user's head orientation from the rendering process [25]. It did this by continually rendering a scene into a cubic environment map. This was then sampled at a high rate, each sample using the latest orientation tracking data. By decoupling the generation of the final image from the slower traditional rendering process, Regan & Pose could minimise apparent latency. Using image composition, they could also combine parts of the scene rendered at different rates, or using different techniques or even hardware. Regan & Pose's implementation was designed to run with a pixel clock of 25.2 MHz, each of the five pipeline stages executing within 40 ns. If each reported stage was atomic, the total latency would be 200 ns. Our system is very similar to that of Regan & Pose, however we use real-time ray casting to perform the sampling address lookup, and we use caching and mip-mapping with lower cost DRAM as opposed to their low latency SRAM for storing the maps.

Mark et al. [16] created an architecture performing 3D post-rendering warping. In this system, a traditional renderer created a set of reference frames containing depth information. One of two possible reconstruction techniques (planar-to-planar warps defining per pixel disparities, or a deformed mesh imposter) could then be used to composit these reference frames into a new image. The reconstruction stage is computationally far simpler than rendering the entire scene, so it allows the system to appear to respond more quickly. Mark et al.'s system through the use of depth information supports changes in both orientation and translation. An exact latency was not reported but Mark et al. state it would be only the time required to perform a 3D warp. Smit et al. [28] pursued this image warping architecture. Their system created a mesh - a grid of vertices with a count equivalent to the reference image resolution. A typical GPU was then used to deform this mesh with a vertex shader implementation of an image warping algorithm. The fragment parameters of the resulting frame could then be used to sample the original image. The image could be reconstructed in a number of ways, however Smit et al. found that treating each vertex as a point and performing screen space point-splatting, with the point size dependent on the depth, provided the best trade-off of speed and quality. They measured the latency of a single-GPU implementation (rendering and then warping on the same device) as 15.7-17.1 ms, depending on scene complexity. The latency of a multi-GPU implementation (one rendering, one warping) was higher at 50.8-57.4 ms, but much less sensitive (lower standard deviation) to scene complexity.

Post-rendering warping techniques have also been augmented with dedicated hardware. Systems such as the Warper Board [29] and WarpEngine architecture [23] implement the warping (or, local loop) stage entirely on dedicated hardware. When combined with a Kalman filter for predictive tracking, the Warper Board reduced apparent latency to 4.33 ms from 42.35 ms. Popescu et al. provided detailed performance models for their system but did not report the total end to end latency. Li et al. performed full depth image warping on an FPGA [14]. They did not report a latency but determined their circuit could run at up to 88.152 MHz. Yanagida et al. [33] proposed using a rate gyro to determine the corrections required for a latent image from the point at which rendering began to right before it was displayed to the user. The image underwent a simple shift and rotation to account for the change in orientation detected by the gyro. The authors used a gyro as it was faster than the absolute magnetic trackers used to render the reference frame. The authors simulated the effect of their system on image quality but did not build the entire system so the latency is unavailable. While our design is a ray-caster, and it does make use of techniques such as mip-mapping and caching to exploit ray-coherence, it has more in common with the designs of Li et al. and Regan & Pose, than hardware-accelerated ray-casters. This is because our design performs no shading and limits the traversal depth to minimise latency, making it more like a lumigraph renderer than a ray-tracer.

One issue with compensating for latency with image warping is that typical techniques use a short history of one or two reference frames with which to compute new images. If the user moves too quickly, or the rate of reference image generation is too low, missing information will result in holes in the warped image. Bergman et al. [1] introduced the concept of a 'golden thread' while describing their adaptive refinement rendering algorithm. The golden thread is a single step that as it is executed results in an ever higher fidelity image. A crude image can be presented to the user quickly, with the quality improved over time where extra rendering time is acceptable. Their renderer approximated this process by supporting a number of rendering modes, of increasing complexity and quality. The renderer would execute each increasingly complex mode, immediately presenting the results to the user, so long as it could before the rendering parameters were updated.

A number of authors have pursued this idea in their construction of frameless renderers, many of which are based around ray-tracing. For example Bishop et al. [2] created a frameless renderer where individual pixels were recomputed using ray-tracing. The pixels to be recomputed were selected randomly. This allowed the image to be updated constantly, independent of the scan-out to the display while avoiding image tearing. Dayal et al. [5] extended this approach, but took advantage of the selective sampling ability of ray-tracing to prioritise updating the most salient parts of the image. They designed an adaptive sampler which would direct the ray-tracer to regions undergoing the most significant changes in space or time. Wooley et al. [32] combine ray-tracing and forward rendering in a very novel progressive rendering approach they term interruptible rendering. They define a unified error metric called dynamic visual error, which encompasses both spatial and temporal error. With this, their system can minimise error by choosing whether to continue to refine a frame, or swap it out for a less detailed, but newer image. Kratz et al. [13] created an adaptive error-controlled sampling controller for a ray-tracing based volume caster. The authors used Finite Element Methods theory to compute the error. The latency of these systems was not reported (and frame rate is a poor analogy in these cases). Petkov & Kaufman [22] present a ray-tracing based adaptive volume renderer, however their sampling controller uses filters, similar to Dayal et al., but in non-cartesian space. The latency of their system was 200 ms when driving a 6 node, 24 screen 7960 x 12088 display. This was a 20x reduction over the best performing frame-based system.

Qu et al. [24] combined 3D warping and ray-tracing in their voxel renderer. The authors' cascaded system performed a 3D warp of a keyframe, and would then fill in any missing information by ray-tracing those specific pixels. The latency was not reported, the authors' test implementation designed evaluate quality rather then speed. Zheng et al. [37] propose a cascaded data path framework with multiple warping stages. Their design is based on the observation that the magnitude of the error introduced in a warping stage, corresponds to the magnitude of the warp. Starting with a frame rendered from a traditional GPU, they chain multiple warping stages, each one simpler than the one before it, but operating at a higher rate. The higher speeds mean the corrections applied by the cruder stages are less significant and therefore less visible, but the perceptual latency is still only as high as the final stage.

Ng et al. [20] also directly coupled the tracker and renderer in their High Performance Touch prototype. This system was designed to emulate a 2D GUI for the purposes of experimentation involving direct-touch user interfaces. The system used a proprietary touch sensor and a Digital Micro-mirror Device projector display, driven by an FPGA, to achieve a latency of ~1 ms.

2.2 Quantifying Visual Quality

When building systems that trade-off spatial quality for speed, authors need a way to quantify the performance of their renderer. As discussed by Ferwerda [7], there are a number of ways to define quality or realism when discussing synthetic images and virtual reality. VR attempts to substitute virtual stimuli for real, and therefore one metric would be sensory believability [12]. Current technology does not have the ability to re-create the full range of visual stimuli that would be required if we were to compare a render to the 'real world' however. Most authors of frameless rendering techniques compare the performance of their renderers to ground truth renders - images produced off-line, simulating an ideal renderer with zero latency. Various metrics are used to describe the fidelity of the first image to the latter, for example Mean-Squared Error (MSE) for static images (e.g. [31]) or Fast Fourier Transforms in the spatial domain for dynamic images (e.g. [34]).

As illustrated by McNamara [17], objective Image Quality Measures (IQMs), like MSE, can deviate significantly from what a user would consider to be a correct characterisation. Accordingly, a number of authors have proposed measures which are based on the operation of the human visual system. These are intended to give more weight to salient artefacts while minimising the influence of (to us) insignificant ones. Wajid et al. [30] compared a number of these measures to determine which could most accurately predict the Image Quality Assessments (IQAs) performed by human participants. They found that MSE & Peak Signal to Noise Ratio (PSNR) were acceptable but Visual Information Fidelity (VIF) ([27]) performed best. However, by considering only those artefacts which are subjectively most significant, we risk missing an unexpected interaction or misrepresenting the fidelity of the image.

For example, motion blur is a rendering artefact which reduces image quality when compared with a ground truth render, but it is also a good visual cue that improves user experience when it stops animation being seen as jerky or strobing [19]. Čadík et al. [4] performed a similar study to Wajid et al. but focusing on the performance of IQMs in assessing artefacts common in synthetic CG imagery, noting that traditional IQMs are often tuned for compression & transmission artefacts. They confirm that no single metric performs steadily for all tested stimuli, though sCorrel (Spearman's Rank Correlation Coefficient over 8x8 pixel blocks) performs relatively well. Zhang & Wandell [36] performed a similar study comparing the simpler Root Mean Squared (RMS), CIELAB and their own spatial-CIELAB ([35]) metrics. They found that the sCIELAB predictions were significantly better than RMS. It is important to consider that various IQMs will have inbuilt biases, and to select a range of measures.

3 Experimental Apparatus

A number of studies have performed objective measurements of the quality of frameless renderers, but so far none have examined what is perceived by the user when these are integrated into a complete system. Below we describe the construction of an experimental system used to render a virtual environment in real-time with ultra low latency, an equivalent GPU-based system, and a configuration that allowed both to be captured and synchronised in order to compare their performance.

3.1 DFE Renderer

To drive the display we implemented a custom renderer on a ANON Dataflow Engine (DFE) [21]. DFEs are processing cards which execute algorithms described as dataflow graphs. The algorithms are implemented as a pipeline of discrete operations laid out in space, rather than time as on a traditional CPU. This spatial parallelism allows each operation to execute with true parallelism, massively increasing the throughput of the implementation. As the pipeline is fixed at design time the latency is known and deterministic. Using this platform we are able to design a renderer with an architecture far different from a traditional GPU and therefore achieve a lower latency.

Our algorithm (or, graph) begins with a set of counters which identify the location on the physical display to be rendered. This location is transformed to undo the distortion of the Head Mounted Display (HMD) lenses. The transformation consists of a per-pixel mapping between a location on the real display and a location on the virtual viewport. The Ray Distortion Sampler Kernel and the Ray Distortion Reader Kernel are responsible for reading the per-pixel disparities from a distortion map, and feeding the locations on the virtual viewport onto the Raycaster Kernel. The Raycaster kernel computes the parameters of sampling rays in the traditional way from camera properties sent asynchronously over PCIe. It then performs intersection tests between these rays and six planes. Each intersection test is a separate series of operations in hardware and the plane parameters are defined at design time. The result of each intersection test is compared with that of the one before it, and the result of the closest intersection is propagated to the next test. Once the closest plane has been identified, the intersection point on its surface, and then a set of UV coordinates are computed. The UV coordinates are converted into a memory address by the Ray Sampler Kernel. This address is sampled by the Ray Sample Reader Kernel. The resulting colour value is combined with timing signals in the Video Signal Generator Kernel and transmitted via DVI to the HMD. A diagram of the dataflow graph is shown in Figure 1.

Both the distortion map and the environment map are stored in DRAM (LMEM). On a DFE multiple DRAM modules are concatenated to form one very wide (1536 bit) address space (LMEM). DRAM is low cost but has high latency and so caching is used to maximise bandwidth utilisation. Both the distortion map and environment map are split into tiles and these are read from DRAM using burst accesses. When LMEM is accessed, upstream kernels generate read commands and downstream kernels read the resulting data. The addressing logic is duplicated in both, so downstream kernels can predict what commands the upstream kernels will send and therefore whether their current cache is valid, and what tiles they can expect to receive subsequently. In a dataflow graph all tokens are executed and transmitted in order. Kernels only run when tokens are available at all inputs and space is available at the output. By duplicating the addressing logic, the upstream and downstream kernels do not have to be explicitly synchronized around the non-deterministic accesses into LMEM. Similarly, because pixels are generated in the same order as they are scanned out to the display, the Video Signal Generator kernel can maintain the current location on the physical display using its own counters, without any direct connection to the first kernel.

The distance across the surface of a plane between two subsequent ray intersection points depends on parameters such as field of view, and distance between the camera and the plane. These cannot be assumed ahead of time and so mip-mapping is used to ensure that for each memory read, on average, 8 subsequent samples can be read from cache. Due to the high latency of DRAM if this were not done the memory would not be able to keep up with the sample requests and the display would be starved of data. The mip level is recomputed for each pixel based on the distance of the current intersection point on the plane, from the previous intersection point.

The design can be scaled to support more than six planes and is limited by space on the FPGA. In addition, transparency mapping is performed by masking collision results based on low resolution 1 bit maps stored in low latency SRAM. Neither of these features are used in the current environment map renderer but can be used to create more dynamic VEs (for example the Pit Room shown in Figure 2.

The dataflow graph generated logically compliant DVI words, which were transmitted from the FPGA via four high speed transceivers. A simple board adapted the physical layer and made the connection to the HMD. The DFE took the place of the GPU in an otherwise typical PC running CentOS 6.5.

3.2 GPU Renderer

In order to compare how the behaviour of our frameless renderer deviated from a typical VE, we constructed an equivalent system, but using

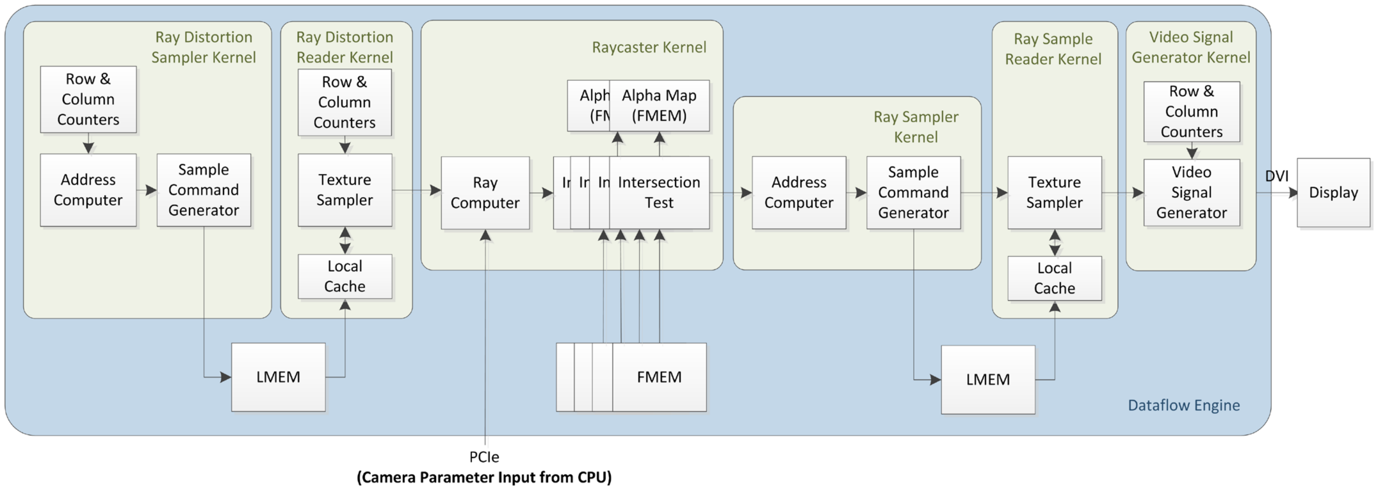

Fig. 1. Diagram of the ray-caster renderer architecture

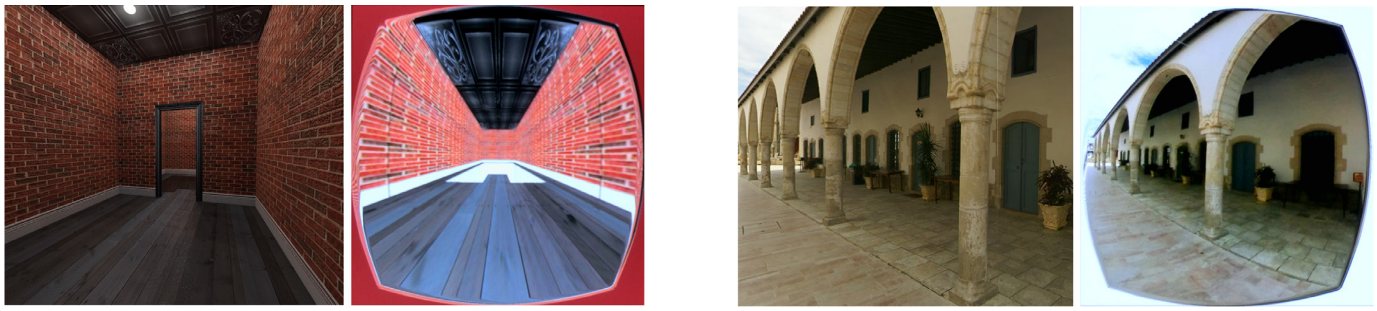

Fig. 2. Images of two virtual environments designed for the DFE frameless renderer, along with photos of the right eye region of the display showing them being drawn in real-time. The Pit Room (left) is a synthetic environment made up of nine alpha-mapped planes. When it is complete, global illumination and shadows will be baked into the textures, as shown in the leftmost image. The Lazarus environment (right) is a six sided cube map captured from the real world, and is the one used in the current experiment.

a GPU in place of the DFE. Our GPU system consisted of a typical PC running Windows 7 with an NVidia GTX 680 GPU. The CPU application was a modified version of OculusRoomTiny, a reference design included in version 0.4.4 of the SDK for our HMD. We modified this application to remove the input processing stages and have it draw a cube made up of six planes surrounding the user's viewpoint. We also removed a feature which used the latest tracking data to adjust the mesh used to apply the lens distortion before post-rendering warping. This was because these types of warps may introduce unpredictable spatial distortions. Further this functionality gives it some features of a 3D warping architecture, and our aim was to make a comparison with a typical GPU system. The use of techniques such as View Bypass and Time Warping which are supported by the SDK can be very effective at reducing apparent latency however, and this is discussed in the conclusion.

3.3 HMD and Tracker

The HMD we used was an Oculus Rift DK2. For capturing head motion, we considered only orientation data. It was captured from the on-board Inertial Measurement Unit Gyroscope, which connects via USB and updates at a rate of 1 kHz. To perform the head motion captures we used the same GPU system described in Section 3.2, however it was modified so that tracker was read in a separate thread not restricted by the GPU. The timestamp of each sample was considered to be the timestamp already assigned by the SDK when it was read in. The mean interval over the entire capture was 1.5 ms, though 22% of the samples had an interval of 1 ms or less. The tracker then is capable of running at 1 kHz, though some variance is introduced between the device and our code receiving the samples. As a result an 18 second sequence is covered by approximately 12100 samples. The DK2 features a 1920x1080 portrait display orientated on its side connected via HDMI. HDMI is backwards compatible with DVI using a passive physical adapter. The screen is split so that each eye sees 960x1080. For our experiment, once the head motion had been captured we dismantled the HMD and secured the display to a camera rig (Figure 3). At this point tracker data was provided from the logs and only the display was used.

3.4 Synchronisation LED

To measure the latency of the rendering stages of our systems, and to synchronise the video captures of the HMD with the tracking data, we needed to be able to instrument the CPU code. To do this we used an Arduino Uno to toggle an LED on command via serial link over USB. To ensure that the latency of the serial link, and rise & fall times of the LED were trivial, we configured the Arduino to loopback all commands, and the CPU code to block until receipt of these echos. The CPU then cycled the LED as fast as it could, while it was monitored with a 1000 fps camera. The total round trip time for two commands was $\sim$3 ms on both Windows and Linux, much shorter than the frame period of rendering captures. The loopback was disabled during captures of the renderers running in real-time.

3.5 Cameras

To measure the latency and confirm correct operation of our apparatus we used a Casio EX-ZR1000 consumer digital camera. This camera is capable of capturing 224x64 video at 1000 fps with a rolling shutter. To capture the rendering systems in operation for image quality analysis we used a PixeLink DL-D722CU-T USB3 camera. This camera had a configurable exposure time down to 1 ms, a global shutter and could capture at up to 257.7 fps. The screen, camera rig and synchronisation LED are shown in Figure 3.

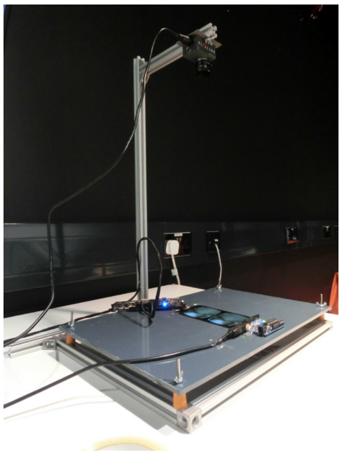
Fig. 3. Image of the apparatus showing the low speed camera, Arduino and DK2 screen

4 Results

4.1 Latency

We measured the latency of the rendering stages of both GPU and DFE systems. To do this we configured the CPU application to cycle between two viewport orientations values, at a rate of 1 Hz. When changing the orientation, the CPU cycled the synchronisation LED to indicate exactly when it had updated the state of the rendering system. 1000 fps video was taken, with both the display and the LED in view. The latency was measured by counting the number of frames between the transition of the LED and the first change in the content of the display (the 'tracker to beam' latency, or, the latency of the rendering stage not including the scan-out time). The technique is similar to those described by He et al. [9] and Di Luca [6].

The GPU system had a latency of 25.7±0.4 ms. This is not unexpected. The display refresh rate is 75 Hz and the CPU application we based our system on syncs its main loop to this. One frame period is 13.2 ms. It will take the CPU one cycle to issue the commands to render, and once complete the GPU will wait for VSync before the frame is swapped to the display. In addition no less than 13.2 ms of latency will be added as the scan moves across the screen. This is because the GPU system scans out a single frame at a time.

The DFE system had a latency typically lower than the the temporal resolution of the video. At this level the synchronisation LED latency becomes non-trivial, so we cannot say the latency of the rendering stage is less than 1 ms. It takes the frameless renderer less time to read a tracker value and update its state than it does to scan out, therefore the latency will be 1 ms at the location of the scan-beam, and 13.2 ms at the location about be overwritten by it.

4.2 Rendering

To better illustrate the differences resulting from the alternate rendering techniques, we applied a simple grid as the texture of our six-sided environment. The CPU was then configured to pitch the camera up and down through 180°at 2-17 Hz, while the display was captured. The DFE system continually updates the tracking data every few lines. The latency is therefore lowest at the point the 'beam' is scanning across the display, and highest at the oldest visible pixel. The frame-based GPU system draws a static image produced for some previous tracking sample. The latency is therefore lowest at the point that scan-out begins (the shortest time between the production of the frame and it beginning to be visible), and increases as the beam moves across the display, as the frame being scanned is ageing while the scan-out proceeds.

If we look at a 13.2 ms exposure capture, we can see clearly the differences in the approaches. The DFE system results in a skewed image under high velocity, never displaying a whole frame for a single tracking sample. The GPU does not have these skewing features, but at the cost of a much greater time between, and therefore difference in, subsequent frames. This is shown in Figure 4. Figure 4 has been annotated showing the latency between regions of the display and the current tracking data (i.e. the ideal tracking data at the time the capture was taken).

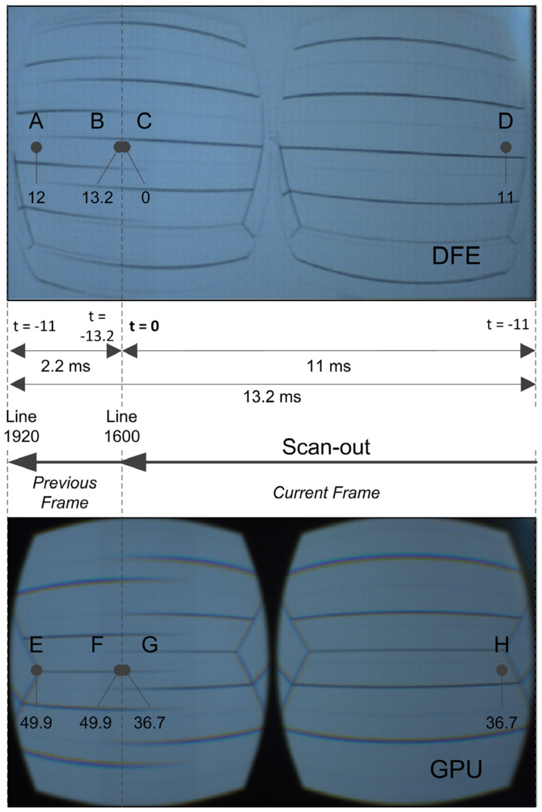

Fig. 4. Captures of the DK2 screen with a 13.2 ms exposure, drawn by the DFE and GPU while in motion, annotated with the latencies at four different locations on each frame for an arbitrary point in time during scan-out (at line 1600)

C The current location of the scan-out. The latency of the DFE is ~1ms. At 144 lines per ms the vertical region around C has the lowest latency of any on the display. This is because pixels drawn around region D, for example, were computed for older tracking samples than those at C.

D The start of the scan-out for the display. This has less meaning for a frameless renderer than a frame-based renderer since the sampling of the tracker is independent of where the scan-out begins.

B,A The scan loops round immediately from A to D, so region B is the oldest region on the frame, last drawn 13.2 ms before the latest tracking data.

H This is where a new frame from the GPU begins. The GPU has a latency of 25.7 ms, so when a scan-out begins the frame is already 25.7 ms old. If we were to capture when the scan was at H, the latency at H would be 25.7 ms.

G As the scan moves across the screen, the 25.7 ms old frame is ageing as it goes. By the time the scan reaches region G, the frame has aged an additional 11 ms. Any part of the display showing the content of that frame (between and including regions H and G) is therefore showing data 36.7 ms old (25.7 + 11), and this latter number will increase as the scan proceeds through F and H. On a V-Synced frame-based system, scan-out always begins at the same place, so these delays across the display will be the same for every frame.

E,F These show the previously rendered frame as they have not yet been overwritten. The previous frame continues to age while the new one is being scanned out, so the latency of any region showing this content is the latency required to complete and draw the previous frame (25.7 + 13.2) plus the time to reach the region in order to overwrite it with the new one (11): (25.7 + 13.2 + 11 = 49.9).

The DK2 uses an Organic Light-Emitting Diode (OLED) display. As an OLED display, the individual pixels have transition times much faster than those of LCDs, closer to that of CRTs. The low persistence of the display is facilitated by rolling scans, in which the screen scans from top to bottom like a CRT, illuminating a narrow moving band of lines as it does so [11]. A 1 ms capture of this is shown in Figure 5. The DFE system should have a visible latency equivalent to the maximum width (in time) of the rolling band, while that for the GPU system will be the time it takes the band to traverse the screen. Since the age of the oldest visible pixel is limited, the skewing features are not so pronounced in the 1 ms capture, although they can be seen with the help of grid-lines.

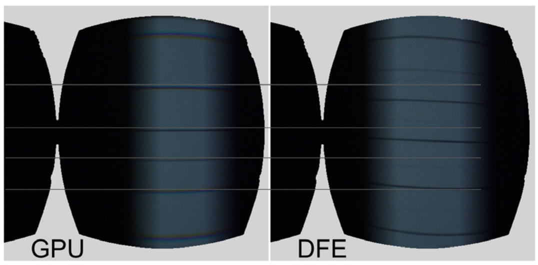

Fig. 5. Captures of the DK2 screen with a 1 ms exposure, with grid lines to help delineate the skew in the DFE render due to motion.

4.3 Image Fidelity Analysis

To assess what effect these differences in rendering approach had on the ability to reproduce virtual stimuli, we subjected captures of the renderers in action to some IQMs. IQMs characterise image degradation. Given a test image and a reference image, they provide an objective numerical score describing how close the two are. Originally designed to measure degradation due to compression artefacts, more advanced IQMs are based on models of human perception, designed so that their scores should correlate with those given by a human. In our experiment we use IQMs to measure the abilities of our rendering systems, comparing their true output with what they would ideally display, if we could build a system with zero latency. The expectation is that the measures will differ between the systems, showing that the rendering technique has a significant effect on what is perceived by the user.

4.3.1 Procedure

While previous studies have incorporated animation ([34]), none compared renders with moving viewpoints. With our renderers however, differences will only be apparent under motion. To create a suitable set of renders, we tracked the head motion of a human participant. We then selected a segment of the capture which had a range of angular velocities consistent with previously observed maximums under voluntary head motions [8]. With this tracking data we were able to produce a set of reference images, and drive the rendering systems in real-time with the same motion.

To measure the quality of the rendering systems' output, we required a set of ground truth reference images. These images are what would be displayed by an 'ideal' VE with zero latency. We produced a set of 18,000 images spaced 1 ms apart in time. Since an ideal system would update the entire display instantly, there was no point in producing images with a temporal resolution beyond that of the captured tracker data. For each image, its timestamp was determined and used to sample the tracker data by retrieving the nearest sample. This sample was used to configure the pose of the camera, then the frame was rendered on the GPU as if it were driving the HMD. Instead of being drawn to the display however the frame was read back from the GPU and written to disk. The result is a sequence of images showing what the HMD should display, if the system had an end-to-end latency of zero and was run at 1 kHz.

The tracking data was then used to drive both rendering systems in real-time, drawing to the screen of the DK2 while it was captured with a camera. The synchronisation LED was used to indicate when in the capture the first tracking data was read into memory. The frame at which the LED transition occurs is considered the epoch, and the timestamp of future frames relative to the first tracking sample are calculated based on the framerate of the capture. The synchronisation LED is cycled by the CPU at 1 Hz to ensure the clocks of the camera and the CPU were matched. Each rendering system was captured twice, once with a 1 ms exposure at a framerate of 257.7 fps, and once with a 13.2 ms exposure at a rate of 75 fps. This is equivalent to the framerate of the DK2 display and approximates a high persistence equivalent of the display. When in focus the DK2 screen and Arduino consumed an area of the frame 1280x600 and the display area of the DK2 screen was 874x491. Examples of the real-time captures are in Figure 6 and an example of a ground truth frame is in Figure 7.

Once the captures had been taken and synchronised, the area outside of the eye view-ports (which is invisible to the wearer of the HMD) was masked and set to black to avoid any luminance changes within it skewing the IQMs. For each tracking sample, the closest (in time) images were selected from the ground truth sequence and real-time capture and these images were compared with a number of IQMs. This was done for all 12,100 tracker samples, and repeated for all four captures (the DFE & GPU with 1 ms exposures (low-persistence) & 13 ms exposures (high-persistence). The IQMs always operate on the entire image. This means the masked area outside of the view-ports will influence the measures but this influence will be constant across all the captures. For the pixels outside of the rolling-band, what is compared with the reference depends on the exposure time. For the 13 ms captures it will be the older pixels which were driven by the band previously, for the 1 ms captures it will be colour of the pixels when they are not being driven.

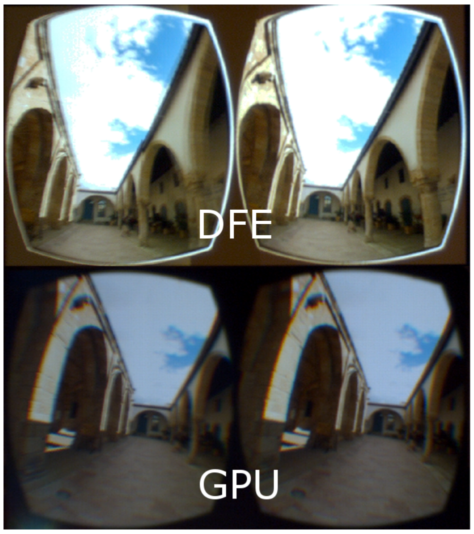

Fig. 6. Example frames from two of the captures of the HMD screen (13 ms exposure)

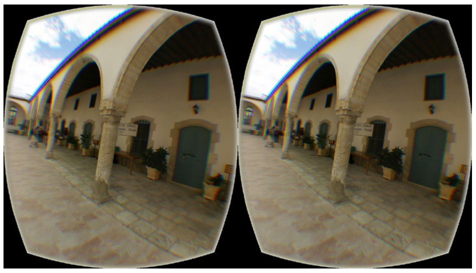

Fig. 7. Example frame from the ground truth renders

4.3.2 Metrics

We cannot compare our stimuli with a real-world ground truth, because we do not have the ability to recreate the full dynamic range that the eye is sensitive to. However, we cannot assume either that the Image Quality Measures (IQMs) designed to mimic human IQAs can identify the optimal stimuli either. We therefore select a range of IQMs, of varying levels of complexity.

- **RMSE** is a measure of the absolute pixel-by-pixel difference of two frames. It is a noisy measure, but simple and fast, and used in a number of previous works.
- **sCorrel (SCOR)** ([4]) performs a Spearmans Rank-Order Correlation on 18x18 pixel blocks. It is a more complex measure than RMSE, but is still not perceptually based. It has the advantage of being less sensitive to brightness changes and low frequency noise, both of which we can expect when comparing images captured with a camera to an offline render.
- **VIF** ([27]) compares the information available in an image to that in its reference. The information is extracted by passing the images through a 'distortion channel' approximating the Human Visual System (HVS).

We pick the above structural metrics because our investigation is geared towards spatial differences caused by scan-out and latency, whereas we can expect large colour discrepencies simply due to differences in the response of the rendering systems, the DK2 screen and the camera. Other frameless renders may pick different metrics that best reveal their differences. Another metric we would have preferred is sCIELAB [35], based on the CIELAB standard (Euclidean distance in L*a*b space). This metric is relatively simple and effective at attenuating low spatial frequency variations [4]. However it requires a mapping of RGB data to real-world wavelengths and dimensions of each pixel. We did not have this calibration for the DK2, nor the inverse through the camera, and so were unable to use it in this study.

4.3.3 Results

From the ground truth to the capture the renders were distorted considerably: by spatial distortions of the lens of the camera, differences between the FOV of the two rendering systems, luminance responses between the two systems and the colour responses of both the display & the camera. How significantly these differences affected the fidelity metrics was dependent on the content of the part of the scene that was visible, and therefore the orientation of the viewport.

This means that the absolute measures are not comparable between two systems, and the effects due to latency will be masked by the effects due to simply pointing the virtual camera in another direction. To ameliorate this, we normalised the error metrics and performed a multiple linear regression on the roll, pitch, yaw and average velocity values of the orientation (predictors) for the measures (responses). The intent being that the velocity coefficient would be mostly free of the influence of the changes due to orientation alone. This is as the velocity correlation will be performed on the residuals after these effects have already been accounted for by the model. The velocity predictor was derived by computing the average of the orientation component angles for each sample, taking the derivative, and then passing through a 10 sample wide smoothing filter.

We performed the linear regression for all captures at both exposures, and the results are shown in Table 1. Only significant predictors $(p < 0.05)$ are shown. Each measure was computed for all tracker samples and so each model has 12100 observations and 12095 degrees of freedom.

Coefficient Estimates and R^2 values for three IMQ multilple linear regression models $(p < 0.05)$

Exposure Time	1 ms		13 ms	
Predictor	DFE	GPU	DFE	GPU
	Normalised Root Mean Square Error			
Roll	0.060	0.063		0.045
Pitch	0.404	0.430	0.210	0.295
Yaw	0.055	0.058	0.031	0.055
Velocity	1.010	1.089	1.008	1.531
R^2	**0.434**	**0.449**	**0.296**	**0.383**
	Spearman's Rank Correlation Coefficient			
Roll	-0.015	0.009	0.084	0.134
Pitch	0.034	0.044	-0.170	-0.201
Yaw	0.010	0.010	0.010	0.021
Velocity	0.039	-0.144	-0.490	-0.802
R^2	**0.192**	**0.215**	**0.436**	**0.481**
	Visual Information Fidelity			
Roll	-0.006		0.058	0.037
Pitch	0.012	0.004	0.031	0.012
Yaw	0.003	0.001	0.022	0.010
Velocity		-0.033	-0.715	-0.410
R^2	**0.155**	**0.142**	**0.236**	**0.108**

Table 1. Parameters for three multiple linear regression models for the RMSE, SCOR and VIF IQMs, showing only coefficient estimates with $p < 0.05$.

An implication of considering each angle element individually for the absolute value, but averaged for velocity, is that the magnitude of the velocity coefficient cannot be compared with that of the angles, as the influence due to orientation will be distributed between these. The purpose of the models are to remove the influence of orientation however, not to examine it. More revealing are the changes in the velocity estimate between the conditions. Considering what we know about how the systems render, we can make four observations that are supported by Table 1.

1. The system with higher latency (GPU), should be more sensitive to velocity, and we see this for almost all cases. The exception is the 1 ms RMSE model, in which we expect the difference in effect is hidden by the noise in the metric.
2. The more complex and sensitive to structure the IQM, the more it should reflect the differences in rendering approach. This is because they should be less sensitive to colour and luminance responses of the screen and camera, which we do not account for. We see this as SCOR and VIF have better fits than RMSE, and larger differences in the coefficients between the GPU and DFE, and 13 ms and 1 ms captures.
3. As the exposure time increases, there will be a higher number of 'older' pixels visible, which will result in a higher average error across the whole screen. This will be exacerbated by high velocities, where the discrepancy of these older pixels will become more egregious. This is the case regardless of the underlying 'tracker to beam' latency, and is reflected in the 13 ms captures having larger coefficients than the 1 ms captures, for both the DFE and GPU.

The DFE should have an advantage in the 13 ms captures however. This is part due to the lower system latency, limiting the age of the oldest pixel, but also because the frameless nature of the display means the age of the oldest visible pixel increases at a constant rate equal to the scan-out time, whereas for the frame-based GPU it increases faster (scan-out time + time since the frame was rendered). The DFE typically has a smaller coefficient for the 13 ms captures, although to what degree this is due to the frameless nature, and what due to the lower average latency, we cannot say.

4. With a 1 ms exposure time, there will be fewer older pixels visible. As a result the error should be less dependent on the average latency across the entire frame, and more dependent on the 'tracker to beam' latency at any given time. This is what we see, with the SCOR and VIF coefficients being smaller for the DFE than the GPU - so far as to be statistically insignificant for the VIF measure.

The results show then that the stimuli produced by systems rendering the same VE, can vary significantly depending on which rendering approach was used under certain conditions (in this case high viewport velocity). IQMs attempt to quantify the perceived difference between two images. The coefficients of velocity are typically smaller for the DFE and low-persistance captures. This implies lower latencies result in higher fidelity VEs under user motion, and that the highest fidelity is provided by the DFE. While the IQMs are perceptually based, it cannot be said that the DFE provides a better experience in absolute terms. To begin with, the comparisons above are done at single points in time, with no consideration of the stimuli before or after. There may be significant temporal interactions with the HVS, which our experiment will not detect. Since measures are not directly comparable between the systems, we also cannot say that one system has generally higher fidelity than the other, only that one is better at approximating the ideal under motion. Further some artefacts may be desirable, such as blurring to reduce the perception of jitter during object movement. Future work will involve using the conclusions here to inform a studies into user performance, to see how the variation in fidelity affects user behaviour in a real system.

4.3.4 Outliers

There is one outlier, and that is the velocity predictor shows a positive correlation with image fidelity for the SCOR metric. Even in a 1 ms capture, where the age of the oldest pixel will be no more than a few milliseconds, the correlation should at best be insignificant. The effect is very small (0.039), but significant (p = 0.0057) and the R^2 is low (0.192). We have no explanation for this result, and can only theorize that it is due to covariance with the orientation. For completeness, the mean covariance for all conditions was $< 1e^{-4}$ for SCOR & VIF, and $< 1e^{-3}$ for RMS.

The DFE estimate for the VIF 13 ms capture is also smaller than that for the GPU, which may on first glance be surprising but should not be considered an outlier. The DFE system maintains its low latency at a cost of image distortion, as it is skewed during scan-out under motion. VIF is the most complex measure and may consider this distortion more egregious than the discrepancies in orientation due to time. When the number of visible (older and distorted) pixels is reduced (in the 1 ms capture) this effect disappears. Future studies may be improved by designing new metrics specifically for frameless renderers, or constraining existing ones only to operate on the visible regions of low persistence displays at a given point in time.

5 Conclusion

In this study we investigated a frameless renderer designed for ultra low latency, but interoperating with an existing, readily available HMD. Our renderer is an implementation of a real-time ray-caster on an FPGA. Ray casting is a highly constrained subset of ray-tracing, making it amenable for hardware acceleration such as we have done. The principle of our renderer is similar to that of Regan & Pose's Address Recalculation Pipeline. Like Regan & Pose, we do not simulate light transport but rely on sampling it from detailed maps rendered off-line. Unlike the Address Recalculation Pipeline however, we use ray-casting with simple geometric proxies. This allows our renderer to operate standalone while still permitting users to translate, as well as rotate, in the virtual world. A 2D image warper in this case would introduce spatial distortions. User's still will not see correct anisotropic visual effects such as specular highlights however - for this our system would need to be coupled to a renderer re-generating the maps in real-time. In this case there would be two latencies, for different visual effects, to consider. This is a similar situation to that encountered with dynamic objects, the behaviour of which may be computed by a loop with latency characteristics quite different to that of the rendering loop. The use of ray casting means our renderer is frameless, in that we can compute pixels in an arbitrary order and minimise the time between tracker data being received, and a pixel being computed. The use of hardware acceleration facilitates an ultra-low latency, which we measured to be ∼1 ms.

Using a series of high and low speed video captures, and objective IQMs, we investigated the implications of combining a frameless renderer with a sequential scan-out OLED display, and compare this with an equivalent system, but built with a GPU. Typically, latency is considered as a discrete characterisation of a constant delay between user input and the response of the display - but this is not complete. Latency changes across the display during scan-out, and is a function of a number of things, including the time to render a single frame, the pipeline depth between the renderer and the display, and the scan-out period itself. Different renderers do not have equal latency responses, and changing the rendering approach can significantly alter this response. For example, we show how the profile of the frameless render is very different to that of a GPU. The latency of the frameless renderer being lowest at the location of the scan-beam, instead of increasing with it from the top of the screen, as with a frame-based system. This is because the frameless system has a 'tracker to beam' latency lower than the frame period, so the more time elapsed since a pixel was driven, the higher the latency of that pixel. Conversely frame-based systems scan out discrete frames, synchronised to the top of the display. As soon as a frame is finished, it is ageing even before scan-out begins, therefore the delay increases as the scan-out proceeds. This difference is manifested as a skew feature under motion on the frameless renderer, as a single scan-out is an amalgamation of multiple tracker samples. Using objective IQMs, we assessed abilities of each system to faithfully recreate a virtual world. Unsurprisingly we found the system with the lowest latency performs better than the high latency system. Further though, we show how the rolling scan approach to low persistence on the DK2's OLED screen reduces the effect of velocity on fidelity. The effect is reduced for both the GPU and the DFE, but on the DFE it is reduced to practical insignificance.

In order to perform the study we disabled the timewarp functionality of the GPU apparatus' CPU application. For other VEs though the Oculus SDK can combine two features to reduce apparent latency significantly. The first, View Bypass, compensates for the rendering latency. The GPU compensates for lens distortion by rendering the scene to a typical (i.e. planar) viewport, then texture mapping the render to a mesh which counteracts the distortion of the lenses. The GPU renders this mesh to the viewport displayed to the user and in doing so applies a post-rendering warp. View Bypass involves re-sampling the tracker right before performing this second render, identifying the change in tracker state since the original frame was rendered, and compensating for it by warping the 2D image and/or the distortion mesh itself. Regardless of the complexity of the original render, the distortion process remains the same for each frame, making the time to complete it highly predictable. This facilitates the second technique, Time Warping, in which the post-rendering warp is left as late possible so it completes just in time for the next scan-out to begin. Under certain conditions this process is very effective and can reduce apparent latency to practically imperceptible levels. It has the disadvantage though of introducing unpredictable spatial distortions. Further, while the perspective may change, dynamic scene content cannot be warped in such a manner. With View Bypass and Time Warping, it is in theory possible to warp on a line by line basis, essentially implementing a frameless renderer. A naive implementation on a GPU however would issue thousands of draw calls per frame (one per line) and the drop in frame rate on any typical PC due to CPU overhead would likely negate any gains. Such a solution is also restricted to per-line warping, whereas a frameless renderer such as ours can update on a per-pixel basis.

Our results were supportive of our observations of the behaviour of the systems, but our experiment had a number of limitations. First, we did not account for the colour responses of the screen or the camera. This lead to high noise floors for the simpler measures such as RMS. Perniciously though, it also meant that the magnitude of the reported errors were dependent on absolute orientation. We used multiple linear regression to minimise this influence, before examining how the render fidelity varied with velocity, although there is no escaping that velocity is in part a function of the orientation.

We did confirm the covariance of the predictors was minimal for our models, but are still unable to explain the outlier described in Section 4.3.4. In expectation of such outliers, and in recognition that the response of participants to visual stimuli is not entirely understood, we chose to use multiple inherently different IQMs. The more sensitive to structure the IQM, the more sensitive to velocity it appeared to be. However this also revealed that in the case of high persistence displays, that for frameless renderers such as ours the skew feature may be more egregious to a participant than the discrepancy due to latency. In the future, a better way to perform such assessments may be to capture the ground truth from the camera & screen. To do this, the CPU application could be modified to draw a static image (pertaining to one tracker sample) to the screen. After enough time for all the pixels to transition has passed, that frame could be captured and considered as what an ideal, in terms of zero latency everywhere, display would show. Capturing the ground truth in such a way would reduce the noise floor to that inherent in the sensor and display itself, removing discrepancies due to colour and slight differences in the geometry transforms.

While we have shown that the frameless renderer has a higher fidelity under motion, how this affects participants in a real VE system is not obvious. One implication of the different rendering techniques for example is the relative latency of the eyes. On a frame-based system one eye will always have a higher latency than the other, whereas one the frameless system the point of lowest latency is constantly changing, meaning the eyes could have on average an equal latency. The use of ray casting, rather than simpler affine transforms, allows our renderer to draw relatively complex virtual environments. The next step will be to put users in a system built with the frameless render to see what effects, if any, there are on presence or other performance measures of a VE.

ACKNOWLEDGMENTS

The authors wish to thank Emil Persson for providing the cubic environment map of Larnaca, Cyprus under CC BY 3.0.

REFERENCES

[1] L. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. *ACM SIGGRAPH Computer Graphics*, 20(4):29–37, 1986.

[2] G. Bishop, H. Fuchs, L. McMillan, and E. J. Scher Zagier. Frameless Rendering: Double Buffering Considered Harmful. In *SIGGRAPH '94 Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 175–176, 1994.

[3] T. J. Buker, D. A. Vincenzi, and J. E. Deaton. The Effect of Apparent Latency on Simulator Sickness While Using a See-Through Helmet-Mounted Display: Reducing Apparent Latency With Predictive Compensation. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 54(2):235–249, jan 2012.

[4] M. Čadík, R. Herzog, R. Mantiuk, K. Myszkowski, and H.-P. Seidel. New Measurements Reveal Weaknesses of Image Quality Metrics in Evaluating Graphics Artifacts. *ACM Transactions on Graphics*, 31(6):Article 147, 2012.

[5] A. Dayal, C. Woolley, B. Watson, and D. Luebke. Adaptive frameless rendering. *ACM SIGGRAPH 2005 Courses*, 2005.

[6] M. Di Luca. New Method to Measure End-to-End Delay of Virtual Reality. *Presence*, 19(6):569–584, dec 2010.

[7] J. A. Ferwerda. Three varieties of realism in computer graphics. In *Proceedings of SPIE Human Vision and Electronic Imaging '03*, pages 290–297, 2003.

[8] G. E. Grossman, R. J. Leigh, L. a. Abel, D. J. Lanska, and S. E. Thurston. Frequency and velocity of rotational head perturbations during locomotion. *Experimental Brain Research*, 70(3):470–476, 1988.

[9] D. He, F. Liu, D. Pape, G. Dawe, and D. Sandin. Video-Based Measurement of System Latency. *International Immersive Projection Technology Workshop*, 2000.

[10] J. F. Hughes, A. van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner, and K. Akeley. *Computer Graphics Principles and Practice*. Addison-Wesley, 3rd edition, 2013.

[11] H. Ito, M. Ogawa, and S. Sunaga. Evaluation of an organic light-emitting diode display for precise visual stimulation. *Journal of Vision*, 13(7):1–21, jun 2013.

[12] H. Kim, T. DiGiacomo, A. Egges, E. Lyard, and S. Garchery. Believable virtual environment: Sensory and perceptual believability. *Workshop on Believability in Virtual Environments*, 2008.

[13] A. Kratz, J. Reininghaus, M. Hadwiger, and I. Hotz. Adaptive Screen-Space Sampling for Volume Ray-Casting. Technical Report February 2011, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2011.

[14] X. Li, B. Liu, and E. Wu. Full Solid Angle Panoramic Viewing by Depth Image Warping on Field Programmable Gate Array. *International Journal of Virtual Reality*, 6(2):69–77, 2007.

[15] I. S. MacKenzie and C. Ware. Lag as a determinant of human performance in interactive systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 488–493, New York, 1993. ACM Press.

[16] W. R. Mark, L. McMillan, and G. Bishop. Post-Rendering 3D Warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 7–16, 1997.

[17] A. Mcnamara. Visual Perception in Realistic Image Synthesis. *Computer Graphics Forum*, 20(4):211–224, 2001.

[18] M. R. Mine. Characterization of end-to-end delays in head-mounted display systems. Technical report, University of North Carolina at Chapel Hill, 1993.

[19] F. Navarro, F. J. Serõn, and D. Gutierrez. Motion blur rendering: State of the art. *Computer Graphics Forum*, 30(1):3–26, 2011.

[20] A. Ng, J. Lepinski, D. Wigdor, S. Sanders, and P. Dietz. Designing for low-latency direct-touch input. *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*, page 453, 2012.

[21] O. Pell and V. Averbukh. Maximum Performance Computing with Dataflow Engines. *Computing in Science & Engineering*, 14(4):98–103, jul 2012.

[22] K. Petkov and A. Kaufman. Frameless Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, XX(XX):1–1, 2015.

[23] V. Popescu, V. Popescu, J. Eyles, J. Eyles, A. Lastra, A. Lastra, J. Steinhurst, J. Steinhurst, N. England, N. England, L. Nyland, and L. Nyland. The warpengine: An architecture for the post-polygonal age. In *Proceedings of SIGGRAPH 2000*, pages 433–442, 2000.

[24] H. Qu, H. Qu, M. Wan, M. Wan, J. Qin, J. Qin, A. Kaufman, and A. Kaufman. Image Based Rendering with Stable Frame Rates. In *Proceedings of the Conference on Visualization '00*, pages 251–258, Salt Lake City, Utah, USA, 2000.

[25] M. Regan and R. Pose. Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques - SIGGRAPH '94*, pages 155–162, New York, New York, USA, 1994. ACM Press.

[26] M. J. P. Regan, G. S. P. Miller, S. M. Rubin, and C. Kogelnik. A real-time low-latency hardware light-field renderer. In *Proceedings of the 26th annual conference on Computer Graphics and Interactive Techniques - SIGGRAPH '99*, pages 287–290, 1999.

[27] H. R. Sheikh and A. C. Bovik. Image information and visual quality. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 15(2):430–444, 2006.

[28] F. Smit, R. van Liere, S. Beck, and B. Froehlich. An Image-Warping Architecture for VR: Low Latency versus Image Quality. In *Proceedings of the 2009 IEEE Virtual Reality Conference - VR 2009*, pages 27–34. Ieee, mar 2009.

[29] D. a. Vincenzi, J. E. Deaton, T. J. Buker, E. L. Blickensderfer, R. Pray, and B. Williams. Mitigation of System Latency in Next Generation Helmet Mounted Display Systems. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 55, pages 2163–2167, sep 2011.

[30] R. Wajid, A. B. Mansoor, and M. Pedersen. A Human Perception Based Performance Evaluation of Image Quality Metrics. pages 303–312, 2014.

[31] B. Watson and D. Luebke. Breaking the Frame: A New Approach to Temporal Sampling. Technical report, 1994.

[32] C. Woolley, D. Luebke, B. Watson, and A. Dayal. Interruptible rendering. In *Proceedings of the 2003 symposium on Interactive 3D graphics - SI3D '03*, page 143, New York, New York, USA, 2003. ACM Press.

[33] Y. Yanagida, M. Inami, and S. Tachi. Improvement of Temporal Quality of HMD for Rotational Motion. In *RO-MAN 1998 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 121–126, 1998.

[34] E. J. S. Zagier. Defining and Refining Frameless Rendering. Technical report, University of North Carolina, Chapel Hill, 1996.

[35] X. Zhang and B. A. Wandell. A spatial extension of CIELAB for digital color image reproduction. *Journal of the Society for Information Display*, 5(1), 1997.

[36] X. Zhang and B. A. Wandell. Color image fidelity metrics evaluated using image distortion maps. *Signal Processing*, 70(3):201–214, 1998.

[37] F. Zheng, T. Whitted, A. Lastra, P. Lincoln, A. State, A. Maimone, and H. Fuchs. Minimizing Latency for Augmented Reality Displays: Frames Considered Harmful. *ISMAR 2014*, pages 1–6, 2014.