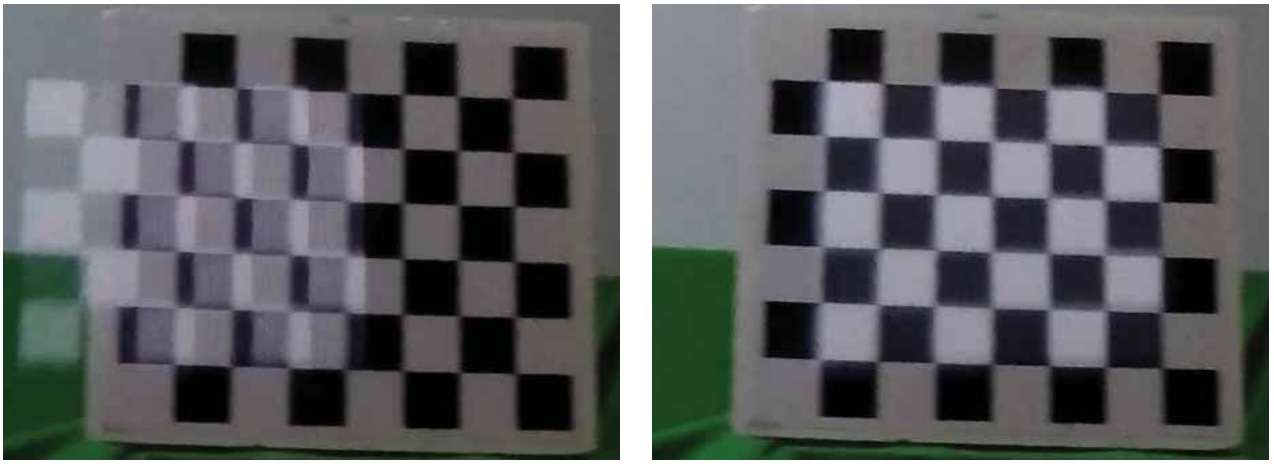
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

From Motion to Photons in 80 Microseconds: Towards Minimal Latency for Virtual and Augmented Reality

Peter Lincoln, *Student Member, IEEE*, Alex Blate, Montek Singh, Turner Whitted, *Member, IEEE*, Andrei State, Anselmo Lastra, and Henry Fuchs, *Life Fellow, IEEE*



(a) Conventional display method.

(b) Our display algorithm.

Fig. 1: A comparison between conventional displays and our latency compensation algorithm at a panning velocity of approximately 50 °/s. In both cases, the motion of the head-mounted display was right to left, causing the off-kilter checkerboard to appear to move left to right. The virtual overlay consists of a smaller checkerboard matched to the central region of the physical board. Note that while the overlay in the conventional display algorithm significantly lags behind the physical checkerboard, our algorithm displays the overlay on top of it and even provides some motion blur when the object moves quickly.

Abstract— We describe an augmented reality, optical see-through display based on a DMD chip with an extremely fast (16 kHz) binary update rate. We combine the techniques of post-rendering 2-D offsets and just-in-time tracking updates with a novel modulation technique for turning binary pixels into perceived gray scale. These processing elements, implemented in an FPGA, are physically mounted along with the optical display elements in a head tracked rig through which users view synthetic imagery superimposed on their real environment. The combination of mechanical tracking at near-zero latency with reconfigurable display processing has given us a measured average of 80 µs of end-to-end latency (from head motion to change in photons from the display) and also a versatile test platform for extremely-low-latency display systems. We have used it to examine the trade-offs between image quality and cost (i.e. power and logical complexity) and have found that quality can be maintained with a fairly simple display modulation scheme.

Index Terms—Augmented reality, latency, display modulation

1 Introduction

Augmented Reality (AR) combines computer-generated virtual imagery with the user's live view of the real environment in real time. An effective persistent illusion that virtual and real objects coexist in the same space requires accurate and stable registration—registration that is both spatially and temporally coherent. However, numerous sources of registration errors are present in most AR systems. Among all error sources, system latency is the largest single source of registration error in existing AR systems, outweighing all others combined [11]. Latency often results in virtual imagery lagging behind or "swimming" around the intended position.

In early graphics systems, latency was expressed as a multiple of the CRT display refresh rate where the multiplier was the depth of the graphics pipeline. Insertion of frame buffers added an additional frame of latency. However, maintaining viewer comfort in today's Virtual Reality (VR) applications makes such latency intolerable. AR requirements are even more stringent because of the need to maintain alignment between real and virtual environments.

A more modern approach to latency reduction is to add a post-rendering "warping" stage through which head tracking position data takes a shortcut past the multi-stage rendering pipeline and modifies the rendered imagery with the most up-to-date tracking data [19, 28]. In addition, the notion of updating displays in raster order as if they were CRTs is fading as a growing variety of display technologies lend themselves to frameless update schemes [7, 10].

- *Peter Lincoln is with UNC-Chapel Hill. E-mail: plincoln@cs.unc.edu.*
- *Alex Blate is with UNC-Chapel Hill. E-mail: blate@cs.unc.edu.*
- *Montek Singh is with UNC-Chapel Hill. E-mail: montek@cs.unc.edu.*
- *Turner Whitted is with UNC-Chapel Hill and NVIDIA. E-mail: jtw@cs.unc.edu.*
- *Andrei State is with UNC-Chapel Hill and InnerOptic Technology, Inc. E-mail: andrei@cs.unc.edu.*
- *Anselmo Lastra is with UNC-Chapel Hill. E-mail: lastra@cs.unc.edu.*
- *Henry Fuchs is with UNC-Chapel Hill. E-mail: fuchs@cs.unc.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

Fig. 2: A user "wearing" our AR Display. The monocular virtual imagery is presented to the user's left eye.

In this work, we have attempted to simplify this modern approach and integrate it into a head-tracked AR display. Our goal is to minimize latency and its detrimental effects in a manner that lends itself to incorporation in a lightweight, low cost, low power, optical-see-through (OST) head-mounted display (HMD) apparatus. We present a combination of simple techniques that accomplish this goal, a description of the apparatus that implements the techniques, and a demonstration of its effectiveness.

Section 2 reviews related work on both VR and AR. Sections 3 and 4 are an explanation of the two parts of our technique along with a description of the physical apparatus used to test it. Quantitative latency analysis and qualitative experimental results are given in Sections 5 and 6 respectively. Lastly, Sections 7 and 8 contain an outline of future work and some conclusions.

2 Related Work

End-to-end system latency is the sum of delays from tracking, application, rendering, scanout, display, and synchronization among components [15, 31]. In AR, latency can cause certain deleterious effects similar to those it induces in VR, such as "simulator sickness" [9]. Livingston and Ai present a user study on registration errors caused by latency, noise, and orientation errors in an AR system with optical see-through head-worn displays [18]. They found that latency was shown to have a significant impact on localization performance, while noise and orientation errors have a limited impact on user performance.

As summarized by Azuma [4] and Holloway [11], existing approaches to reducing dynamic registration errors caused by latency can be classified into four categories: latency minimization [23, 26], just-in-time capture and processing [14], specifically also image generation [21, 24], predictive tracking [3], and video delay [5]. The latter method delays the video stream to be augmented such that it is synchronized with the corresponding tracking information (including any "closed-loop" tracking obtained from the video imagery itself). While it can completely eliminate relative latency between background imagery and augmentations, this is accomplished at the expense of delaying the entire augmented (composite) image relative to "reality."

In current head-worn display systems, delay caused by waiting for vertical sync can be the largest source of system delay [15]. Recently, Nvidia introduced G-Sync™ to maximize input responses by making the display accept frames as soon as the GPU has rendered them [22]. AMD introduced a similar but license-free solution called FreeSync™ [1], which is built upon the DisplayPort™ 1.2a standard.

Post-rendering updates were an outgrowth of image-based rendering and include such schemes by McMillan and Bishop [20], Regan and Pose [25], and Shade et al. [27]. Integration into more conventional rendering pipelines has taken several forms as seen in Talisman [16], the render cache [30], and adaptive frameless rendering [10]. While originally introduced as a means of reducing rendering cost via reuse of previously rendered pixels, their utility in reducing rendering latency has brought them into the mainstream of VR and AR. Smit et al., demonstrate a system that achieves image generation and display at the refresh rate of the display using a client-server, multi-GPU, depth-image warping architecture [28, 29]. The client on one GPU generates new application frames at its own frame rate depending on the scene complexity, while the server on the other GPU performs constant-frame-rate image warping of the most recent application frames based on the latest tracking data.

The evolution of displays has accelerated, with LCDs replacing CRTs and newer technologies appearing in projectors and small displays. For interactive applications, digital micromirror display (DMD) projectors based on microelectromechanical systems (MEMS) mirrors offer refresh rates in the range of tens of kHz, but are basically binary displays requiring additional modulation processes to yield perceived gray levels [12]. Zheng et al. [31] present background on how DMDs operate and can be used in low latency displays. Similarly, active matrix organic LED (AMOLED) panels are increasing in popularity, especially in mobile applications, because they are both potentially less expensive and more efficient than LCD displays. While AMOLEDs are inherently analog devices suitable for directly displaying gray scale pixels, they have electrical properties that make binary mode appealing [13]. Both DMDs and AMOLEDs exhibit rapid refresh rates and are therefore leading candidates for low latency VR applications. With suitable partially reflective optics both are applicable to AR.

Lately, post-rendering update has been combined with frameless, non-raster display refresh as a means of latency reduction [31], albeit in non-head-tracked configuration using precomputed content. The work described in this paper is an extension of that idea implemented in a head-tracked display with real-time update processing and a simplified display driver embedded in the head-mounted device.

3 Low Latency Rendering Approach

As presented in Figure 3, we employ a combination of traditional PC/GPU rendering with a post-rendering FPGA-based display update process. Synthetic elements of the AR scene are rendered on the PC, using an NVIDIA® GeForce® GTX™ Titan Black GPU at 60 Hz and transmitted via DisplayPort™ to the display processing system. That system consists of a Xilinx Virtex-7 (XC7VX690T-2FFG1761) FPGA board (HTG-777[1]) interfaced to a Texas Instruments DLP® Discovery™ 4100 Development Kit[2], which is composed of a Xilinx Virtex-5 (XC5VLX50-1FF1153) FPGA board, an XGA (1024×768) DMD module, and some rear-projective illumination and optics. The display system is capable of displaying binary frames to the user at up to 16 kHz. In order to track the motion of the HMD, we use high resolution rotary shaft encoders (US Digital E6 Series), each with 40,000 ticks of resolution per revolution, though this limits the HMD to only angular motion. This "open-loop" tracking system, using quadrature based encoding, is processed by a Spartan-III FPGA board (Digilent) and routed both to the render PC and the display processing system using RS-232 serial links. Each device receives tracking updates at a rate in excess of its display update rates. The entire assembly, including FPGAs, DMD, and optics is presented in Figure 4.

The PC performs standard AR scene rendering from the user's (or test camera's) tracked and calibrated viewpoint, and warps the scene so that the middle 1024×768 region of the 1920×1080 image matches the real viewpoint for the most recent tracking update that the PC has received. We used OpenCV's camera calibration library to perform the viewpoint calibration. The rest of the image contains valid imagery that would be appropriate if the display's field-of-view were larger. If

[1] http://www.hitechglobal.com/Boards/Virtex-7_FMC.htm
[2] http://www.ti.com/tool/dlpd4x00kit

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

Table 1: Comparison of Modulation Technique Requirements and Behavior

Method	*Requirements per Binary Pixel*							*Major Oscillation Frequency*	*Response to Input*
	Memory Operations			*Math Operations*					
	Desired	*Accumulation*	*History*	*Comparator*	*Add/Sub.*	*Counter*	*LFSR*		
PWM (Counter)	Read			Yes		Yes		Low	Slow
PWM (Zheng et al.)	Read	Read/Write	Read/Write	Yes	Yes, 2			Low	Slow
PDM (Delta-Sigma)	Read	Read/Write		Yes	Yes, 2			High	Fast
PR-PDM	Read			Yes			Yes	High	Fast

the HMD were not moving, then without any additional processing, simply displaying the middle region of the received image would optically register with the real world for a coherent scene. However, when the user begins moving, then the illusion would be destroyed by the latency along the tracker-PC-GPU-scanout-display datapath.

In order to achieve low end-to-end latency, we divide the rendering workload between the PC and the display processor. As a side channel in the rendered image, we include the then-current tracking information used to render that image. Independently, the display processor receives the live tracking information directly from the tracking system. When it is time to begin displaying a 16 kHz binary frame, we use both the viewpoint calibration and difference in the two tracking states to transform the latest image to the current viewpoint. Essentially, the display processor is performing a post-rendering (and post-transmission) warp. Some sample simulated frames of the imagery in this process are presented in Figure 5.

In order to simplify the processing required on the FPGA for this warp, we make some assumptions about the motion. Our tracking system only supports motion along two rotational dimensions (pitch and yaw), and the center of projection of the moving viewpoint is near to those two rotation axes. We also assume that the difference in angular pose between the render-time pose and the live pose is small. Thus we can simplify our warp into a 2D translation and crop of the CPU-supplied imagery; essentially we select a particular 1024×768 window of the full rendered image for each output display frame. As a result, that "excess" resolution provided by the GPU becomes padding for the offset computation engine. For each dimension, the offset (Δ) of this window in the input frame is simply the product of the difference of the rotation angles (ω) with the ratio of display pixel resolution (r) to field-of-view angle (α): $\Delta = (\omega_{live} - \omega_{render}) \times (r/\alpha)$. In the event that the user managed to move faster than the resolution provides, out-of-bounds pixels are replaced by black, though we did not encounter this in our live tests.

The Virtex-7 FPGA receives the DisplayPort-provided frames at full resolution and stores them in a double-buffer of 2048×1536, dual-ported (simultaneous reads and writes at different clock rates) memories with 6-bit graylevel resolution. While we currently receive frames at a resolution less than the storable size, we selected a power-of-two width in order to support the highly parallelized modulation engine (see Section 4) since it requires reading 128 pixels simultaneously each clock cycle. In order to fit the pair of images in fast internal FPGA SRAM, we had to limit the graylevel resolution to 6 bits/pixel, which we found visually sufficient. By using a double-buffered framebuffer in the FPGA, we are generally able to read from a buffer with a consistent tracking state. It is occasionally possible that the writing process may have switched buffers just after the reading process decided on which buffer to read from, but because the reading process (about 16 kHz) is orders of magnitude faster than the writing process (60 Hz), the discrepancy would last for only a small portion of one binary frame. If we were to use only a single framebuffer, then the tracking state would not be consistent across the whole image while the writer was modifying it, which would affect many binary frames. Using two framebuffers to store the desired imagery keeps the reading algorithm and address offset computation simpler, at the expense of doubling the memory required for storing desired imagery.

4 Display Modulation Approach

A key contribution of our work is a novel approach to modulation of the DMD projector's binary output to approximate desired grayscale images. We begin this section with a review of two classical modulation approaches: pulse width modulation (PWM) and pulse density modulation (PDM). We then describe the approach introduced by Zheng et al. [31], which sought to take advantage of the human eye's response time, but in practice suffers from excessive flicker and requires significantly more memory, proportional to the integration window length. Finally, we describe our approach that deliberately introduces randomness into the modulation, thereby providing two crucial benefits: eliminating output flicker and yielding a much simpler hardware implementation. A summary of a comparison of the modulation techniques we discuss is presented in Table 1.

4.1 Background: Classic Modulation Schemes

The use of a DMD requires conversion of grayscale pixel values to a sequence of binary frames that approximate those values. This is a form of analog-digital-analog conversion: each "analog" pixel value is actually a 6-bit grayscale value, which is converted to a series of 1's and 0's that represent white and black projected values, which in turn, due to persistence in the human visual system, the observer sees as an analog grayscale value.

That is, to display a given 6-bit intensity k over a period of 64 bit-times, the light is turned "on" for k bit-times and "off" for the remaining times. The "on" and "off" pulses are integrated (in this case, by the human eye), and result in the appearance of the desired intensity. The differences between the schemes presented below are in the selection of which k bit-times should be "on". The selection algorithm affects the perceived quality of the resulting image (e.g., flicker) and determines the requisite hardware resources (memory, memory bandwidth, and computation time).

There are two well-known approaches for converting analog signals to bitstreams (i.e., a sequence of 1's and 0's): pulse width modulation (PWM) and pulse density modulation (PDM). In both approaches, the pulse generator operates at a high enough frequency so that the moving average closely approximates the analog value, when averaged over a window of time smaller than the human eye's response time. We call this window the integration interval.

In PWM, the width of the pulse of light generated is varied in direct proportion to the desired gray value at the pixel, as illustrated in Figure 6(a)(b)(c). Thus, a 25% gray value will generate a pulse that is "on" for 25% of the time and "off" for the remaining 75% of the time window.

In PDM, on the other hand, each pulse is of a fixed width but the density of "on" pulses is varied in direct proportion to the desired gray value. Thus, a 10% gray value will generate one 'on' pulse followed by nine "off" pulses, in a repeating pattern, with each pulse only a few microseconds in duration. In effect, PDM is equivalent to a much higher frequency PWM, resulting in output that is much smoother in time.

The ideal method for generating PDM pulses is known as delta-sigma modulation [2], illustrated in Figure 6(d)(e)(f): an "on" pulse is generated if the accumulation of all the prior "on" pulses ("sigma") is an entire unit ("delta") less than the integration of the desired gray value over that time. While delta-sigma is a very powerful approach

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

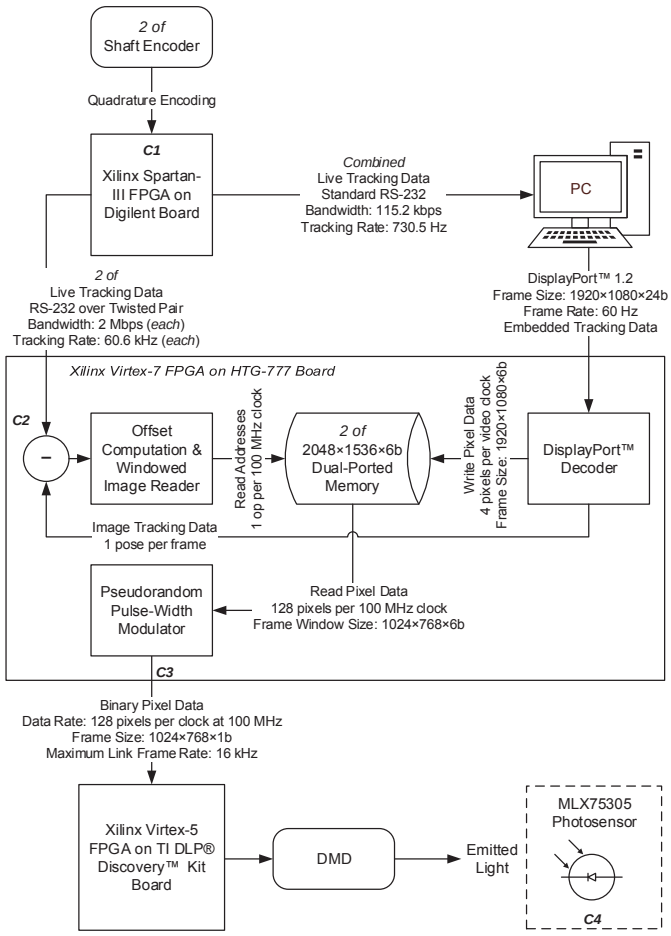


Fig. 3: Implemented data path framework. Labels *C1-C4* indicate oscilloscope channels assigned to measurement test points used during latency analysis. The photosensor (*C4*) is present and used only during latency testing.

with many variants (e.g., higher-order integrators, continuous- vs. discrete-time, etc.), for the specific case of fixed-rate binary values, it is equivalent to Bresenham's line drawing algorithm [8]: an increment in x is accompanied by a change in y only if the deviation from the desired line would be more than one unit. Addressing our use case specifically, implementing delta-sigma would require at least as much memory as storing one desired image, since an error term must be maintained for each pixel.

There are other well-known modulation approaches as well—e.g., pulse code modulation (PCM) and pulse frequency modulation (PFM)—but they are not suitable for our purposes. In particular, PCM converts each analog value to a multibit digital value, not a binary sequence needed by the DMD projector. Similarly, PFM requires the frequency of output pulses to be continuously variable, which is again a poor match for the constant frame rate of the projector.

4.2 Prior Work: Approach of Zheng et al.

Zheng et al. introduced a modulation approach that was based on PDM but incorporated a model of the human eye's response in an attempt to pre-correct for visual response delays [31]. In particular, their approach assumes that the human vision remembers the average of the past few frame values over a sliding window of time (e.g., 64 binary frames). Thus, a new binary value is produced with the assumption that the previous 63 binary values are still contributing to the eye's response.

While the idea of incorporating the eye's response delay into PDM is an interesting one, in practice it often leads to unseemly flicker. For example, suppose the gray value of a pixel has just changed from 100% to 50%. The history of previous binary frames will all have a "1" value; therefore, the next binary frame will have a "0" value for this pixel. In fact, for the next 32 binary frames, a "0" value will be generated because the sliding window of 64 frames has a majority of "1" values. The result is an alternating sequence of 32 "on" frames, followed by 32 "off" frames, and this pattern repeats, resulting in significantly choppy output that is closer to PWM than PDM. In effect, the attempt to account for the eye's memory has resulted in adding feedback into the computation, thereby causing highly undesirable oscillatory behavior.

Thus, while the idea of incorporating the eye's response in this way may be a good one in principle, it can produce excessive output flicker in practice, which is a significant drawback of their approach.

4.3 Our Approach: Pseudorandom Pulse Density Modulation

We base our approach on PDM, but deliberately introduce randomness into the modulation; we call this Pseudorandom Pulse Density Modulation (PR-PDM). The goal was to eliminate the output flicker of PWM-like approaches (including Zheng et al. [31]), yet to obtain an implementation complexity much less than that of ideal PDM (i.e., delta-sigma modulation). While we assume that a human eye would integrate over time, we don't explicitly model for it, but instead target flicker frequencies that are not human perceptible.

The key idea is that for each binary frame, a pixel's output value is chosen to be a "1" or a "0" randomly using a probability proportional to its gray value. In particular, we use a hardware-implemented pseudorandom number generator, using the well-known linear-feedback shift register approach (LFSR) [17], to generate 6-bit pseudorandom numbers from 0 to 62. For any binary frame, if a certain pixel's gray value, normalized to the range [0, 63], is greater than the pseudorandom sample for that frame, then a "1" is output, else "0." Thus, over a window of 63 frames, the number of 1's will approximate the pixel's grayscale value. In fact, if the grayscale values are represented using 6-bit integers, then the number of 1's will exactly equal the grayscale value. The randomness of the 1's and 0's is akin to dithering in time, thereby eliminating the choppiness of Zheng et al.'s approach [31]. An example sequence of PR-PDM output pulses is illustrated in Figure 6(g)(h)(i), and a sample capture of the output pulses measured by the light sensor are illustrated in Figure 7. Some sample binary frames produced through this process are presented in Figure 5.

Somewhat unexpectedly, introducing randomness into the modulation approach also has a significant implementation benefit: rather than increasing design complexity, it actually results in a much more efficient implementation than Zheng et al.'s [31] or classic delta-sigma modulation.

In particular, Zheng et al.'s approach must store the entire history of the most recent 64 frames (i.e., values over the entire integration interval) so that it can compute the accumulated value over a sliding window in time. Moreover, for each new binary frame, a pixel's computation requires several memory accesses: reading the desired grayscale value, reading the current accumulated value, reading the old binary value from 64 frames prior (so it can be subtracted from the windowed accumulation), storing the new accumulated value, and storing the new computed binary value. All of this imposes a large requirement on memory bandwidth, thereby limiting the frame rate at which the system can be operated and, equivalently, increasing latency.

In contrast, our approach only requires a single memory access for a computation. The desired grayscale value is read from memory, compared with a random number, and a binary output is generated. Thus, the demand on memory bandwidth is dramatically lower, allowing us to achieve a significantly higher frame rate of 15,552 frames/sec. For 6-bit graylevels, and double-buffering, we use about 30% of the memory bandwidth and less than 15% of the memory storage that Zheng et al.'s approach would require. Keeping our memory usage low enables us to use fast FPGA SRAM. External DRAM, while being able to store all of the data of any of the discussed algorithms, would have lower throughput and restrict the frame rate accordingly.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

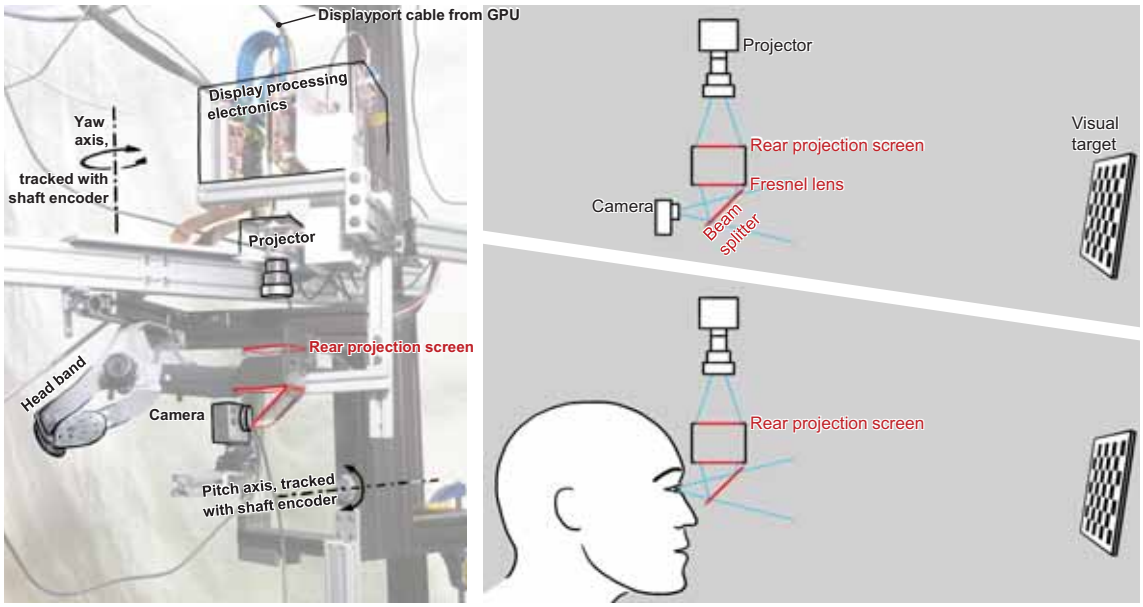


Fig. 4: System Assembly. Display processing electronics components include the TI DLP® Discovery™ 4100 Kit (Virtex-5 FPGA), a HTG-777 FPGA board (Virtex-7 FPGA), and a custom DisplayPort input and interconnect board. Projector components include the XGA DMD chip and standard lens assembly. The remaining optics (red outlines) include the rear projection screen, Fresnel lens, and beam splitter (half-silvered mirror). Either a user (see Figure 2) or a camera can observe the optically combined physical and virtual visual targets.

Our randomized approach is also simpler to implement than pure delta-sigma modulation. The latter requires two memory reads and one store per computation: reading the desired grayscale value, reading the accumulated value, and storing the updated accumulated value. Thus, for double-buffering, our approach requires only one-third the memory bandwidth and only two-thirds of the memory storage of delta-sigma.

In summary, our approach of introducing randomness into modulation provides two crucial benefits: no perceptible output flicker and a much simpler hardware implementation.

4.4 Perceptual Comparison of PWM and PR-PDM

The perceptual advantage of PR-PDM over PWM can be understood in terms of the noise spectrum of the modulated light output; in short, PR-PDM moves modulation noise to higher frequencies, eliminating perceptible flicker. Referring to Figure 7(a), showing light intensity over time for a 25%-gray value, we see that the light alternates between "on" and "off" and is, on average, "on" 25% of the time. If we had continuous control over the light intensity (e.g., if we could control the current delivered to the illuminator), then we could directly-generate a 25% value and the intensity value would be a straight line (a DC value) at 25% of the amplitude of the signal shown in the image. Any deviation from this DC value (the desired signal) can be thought of as noise; in the present case, this noise is due to the modulation scheme. If enough of the modulation noise is at a perceptible (low) frequency, then we will see flicker. The advantage of PR-PDM over PWM is that PR-PDM moves the noise power to higher frequencies.

To demonstrate the noise-spectral difference between these modulation schemes, we performed a simulation in LTspice[3] wherein a DC signal, at 25% of full-scale, is modulated by 6-bit PR-PDM and PWM at a 16 kHz sample frequency. The simulation was carefully set up to be faithful to the PR-PDM modulation scheme implemented in the actual display: the simulated PR-PDM output in Figure 8(a) is identical to the output in Figure 7(a) captured on the actual display. Figure 8(b) shows the same signals passed through an RC low-pass filter with a pass band of 125 Hz (chosen arbitrarily for illustrative purposes). While the two waveforms (PR-PDM and PWM) produce the same average value (about 1.25 V), it is clear that the PR-PDM output has a much smaller swing around the average, and therefore much less noise overall, when compared to PWM.

The distinction between these methods is easier to visualize in the frequency domain. Figure 9 is a plot of the spectra of the modulated, unfiltered signals shown in Figure 8(a). We see that both the PWM and PR-PDM have significant power at 254 Hz and harmonics thereof. Notice, however, that the power at this frequency in PWM is about 15 dB (32 times) higher than PR-PDM. It is apparent that the noise introduced by PR-PDM is contained primarily at frequencies at or above 1 kHz (and is thus far less likely to be perceived). (In the ideal amplitude modulated case, the output would only have a DC term.)

5 Motion-to-Photon Latency Analysis

The overarching aim of the present research is to minimize the delay between a change in the user's position or orientation and the corresponding change to what is seen on the display. Our system has been carefully instrumented such that this delay can be measured precisely and such that sources of residual latency are identified and well characterized. The discussion that follows is focused solely on latency analysis and explicitly omits other details that are covered elsewhere in the paper.

5.1 Signal Path

In our apparatus, head motion is detected by means of an optical rotary encoder with a resolution of 40,000 ticks per revolution (one tick is 0.009°). Considering the resolution and field of view of our display device, a rotation of about 3.45 ticks corresponds to a single pixel or, equivalently, the worst-case error in rotation measurement is about 0.29 pixels.

As shown in Figure 3, the rotary encoder emits quadrature-encoded data which is decoded by an FPGA (separate from the display control FPGA). The present angle is transmitted continuously to the display control FPGA via a serial link. The angle is received by the display control FPGA, where it is used to compute the correction to be applied to the displayed image. This computation takes about 4 µs and is performed between binary frames. Importantly, the correction is performed once per binary frame using the most recently received angle.

Binary pixel data is streamed to the TI DLP® Discovery™ Kit's FPGA processor in four "quads," each comprising one quarter of the

[3] http://www.linear.com/designtools/software/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

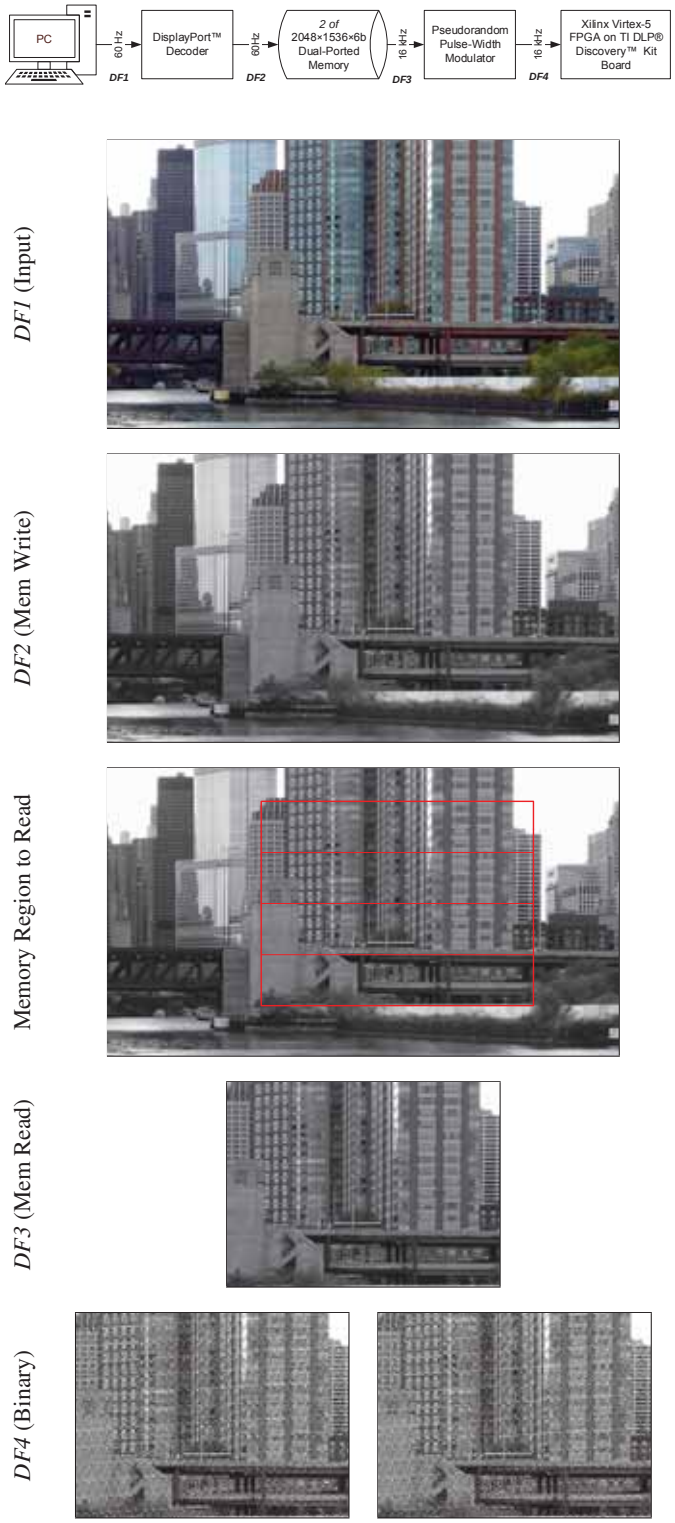


Fig. 5: A sample dataflow. The flow diagram at the top is a subset of the diagram in Figure 3 showing the video pathway. *DF1-4* indicate the four raster-order image streams and correspond to the rows labels. *DF1* and *DF2* operate on the video clock, which stores images into double-buffered memory at a 60 Hz frame rate, alternating buffers with each input frame. Independently, the read process (*DF3*) selects a window in the inactive buffer (the buffer not currently being written) in the memory based on the difference in the tracking data. This window is shown as a red outline, subdivided into the four mirror-reset regions of the DMD. When the poses differ (as shown, e.g. live viewpoint is up and to the right of the render-time viewpoint), then a non-center offset region is picked. Two subsequent binary frames (*DF4*) are shown for the same input poses.

rows of mirrors on the DMD. After each quad is transmitted, a "commit" signal is sent and, after a short delay, the respective set of mirrors begins to move as commanded.

Finally, in order to detect the photons and measure the total latency, a precision light sensor, based upon the Melexis MLX75305 light-to-voltage IC, was temporarily fitted onto the focal plane of the HMD.This sensor emits an analog voltage signal proportional to the luminous flux falling upon it. We measured the -3 dB bandwidth of this sensor at about 55 kHz and the device is specified with a rise time of 6 µs. An opaque barrier is installed above the sensor with an approximately 0.25 mm pinhole located just above the light-sensitive element in the IC; this ensures very directional sensitivity and reduces the effect of ambient light sources.

5.2 Measurement Instrumentation

To measure motion-to-photon latency, the system was configured such that it either displayed zero intensity or full intensity, depending on the angle of the HMD. We will refer to the angle at which the display changes from zero to full intensity as the trigger angle; the display shows black when positioned anywhere on one side of the trigger angle and white on the opposite side.

The apparatus is instrumented such that signals can be monitored at four critical points in the resulting signal path (see Figure 3 for the datapath location of these test signals):

1. *Motion Initiated*: The quadrature-decoding FPGA drives a pin high or low depending on the HMD orientation with respect to the trigger point. The delay between the physical movement and this signal is on the order of 100 ns. This signal is for instrumentation purposes only and is not in any way coupled to the display control.

2. *Data Received*: The display control FPGA drives a pin high or low based on the angle received over the serial link in relation to the trigger angle. This signal also corresponds to the new angle being latched into the correction-computation circuitry.

3. *Pixel Transmitted*: The value of the bit of the first pixel of each chunk of data streamed to the DLP Kit's processor is mirrored to a pin on the control FPGA. In particular, if zero-intensity pixels are being sent, this line is low and if full-intensity pixels are being sent, this line is high. Thus, we observe the transmission of each quad whenever full-intensity is being displayed. This signal is also low when no data is being sent.

4. *Light Emitted*: The analog voltage from the light sensor, positioned on the focal plane of the HMD, is monitored. The sensor is located such that its aperture sees the first quad of the image.

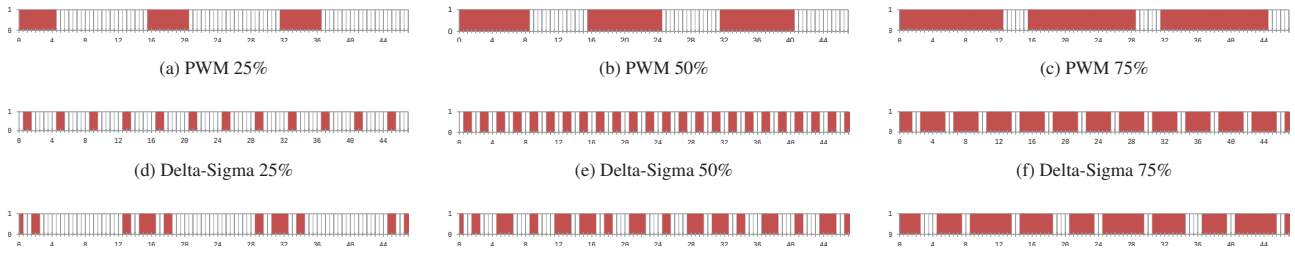
We simultaneously probed these four signals using a Tektronix TDS 684B oscilloscope (1 GHz analog bandwidth at 5 gigasamples per second). The scope was configured to trigger on the rising edge of signal (1), above, i.e., when the HMD crosses the trigger angle (in this case resulting in the display switching from black to white).

5.3 Latency Component Analysis

An example of a resulting trace is shown in Figure 10; channels 1-4 correspond to the respective signals described above. In this figure, the time base is set at 20 µs per division. We will use this trace to step through the characterization of the latency components in the present system.

Transmission of Position Data Referring to channels 1 and 2 (green and blue, respectively), we observe a delay of about 27.5 µs between the motion event and receipt of the new angle by the display processor. This delay is due to the time taken for the angle information to be fully transmitted via the serial link. In the present implementation, the serial link runs at 2 Mbps; the transmission of the angle data plus serial framing overhead takes approximately 15 µs. The observed delay indicates that the crossing of the trigger angle occurred just after the quadrature-decoding FPGA had begun transmitting a previous

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

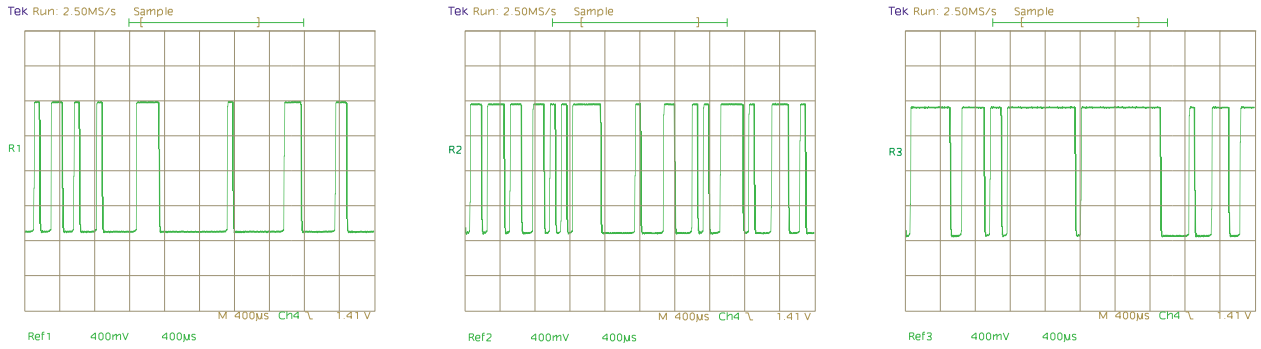


(a) PWM 25% (b) PWM 50% (c) PWM 75%

(d) Delta-Sigma 25% (e) Delta-Sigma 50% (f) Delta-Sigma 75%

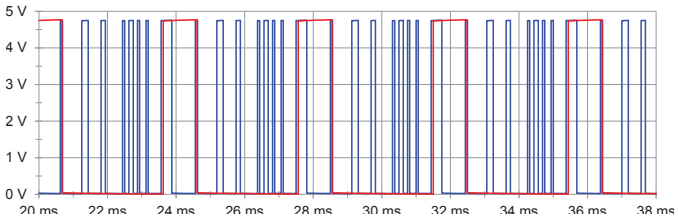
(g) PR-PDM 25% (h) PR-PDM 50% (i) PR-PDM 75%

Fig. 6: Sample comparison of simulated 4-bit graylevel modulation schemes by method and desired constant intensity level. Red indicates on, and white indicates off.

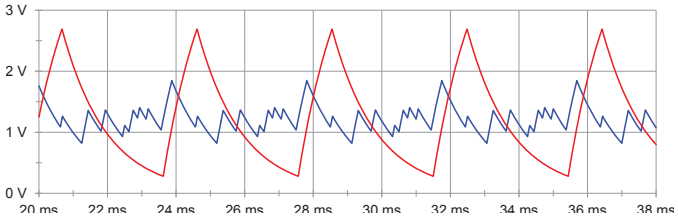


(a) PR-PDM 25% (b) PR-PDM 50% (c) PR-PDM 75%

Fig. 7: Measured 6-bit graylevel modulation patterns for PR-PDM by desired intensity level using the light sensor. A high voltage indicates the pixels are "on," and a low voltage indicates the pixels are "off."



(a) Unfiltered signals.



(b) Signals in (a) with RC low-pass filter with a pass band of 125 Hz applied

Fig. 8: Simulated signals of 6-bit PWM (red) and PR-PDM (blue) for targeting a constant DC signal of 25% intensity.

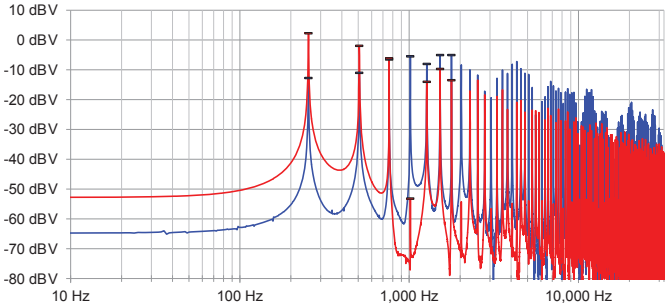


Fig. 9: Spectra of 6-bit PWM (red) and PR-PDM (blue) simulated signals from Figure 8(a). For clarity, the local maximums of the first several harmonics are highlighted with dash marks of the same color.

value. In general, with the present implementation, this delay component varies between 15 µs and 30 µs (22.5 µs average), depending on the precise moment the motion occurs. The magnitude of this delay is directly proportional to the bandwidth and latency between the tracking device and the display controller. Ideally, these devices would be closely coupled, bringing this latency term close to zero.

Binary Frame Phase Referring to channels 2 and 3 (blue and red, respectively), we observe a delay of about 46.8 µs between receipt of the trigger angle and the beginning of the transmission of the full-intensity data to the DLP Kit. This delay is substantially due to the new angle arriving during the transmission of the previous binary frame (which is zero intensity and thus does not show on the red trace). Recall that the new angle cannot be applied until the next binary frame. Binary frame transmission takes approximately 70 µs, so this delay will range from zero to 70 µs, with an average of 35 µs. This delay is inherent due to the binary frame rate (which is about 15,552 Hz in the present implementation). A higher frame-rate device would result in a proportionally smaller value for this latency component.

Mirror Refresh Referring to channels 3 and 4 (red and yellow, respectively), we see that the light sensor begins registering increased intensity just after the first quad of the binary frame has been committed (seen as the short dip in the red signal). This is as expected due to the operation of the DLP device. The 6 ns to 8 ns rise time in the light level is most probably due to variances in how fast the mirrors switch combined with the rated response time of the sensor itself. We declare photons to have arrived when this signal has risen about 50%. This time interval is inherent to the DLP device (operating at this frame

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

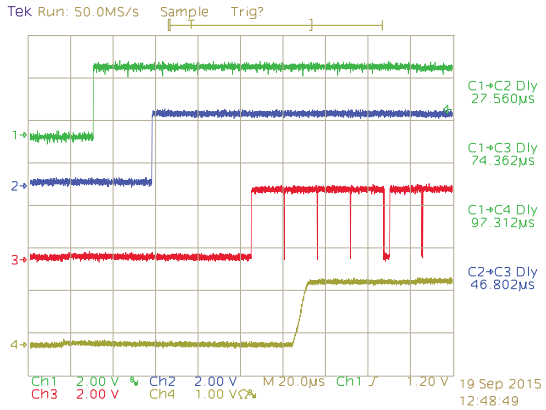


Fig. 10: Sample motion-to-photon latency measurement. Channels from top to bottom: (1, green) motion initiated, (2, blue) data received by display processor, (3, red) pixel transmitted, and (4, yellow) light emitted. In this example, total latency (C1→C4) was 97.312 µs.

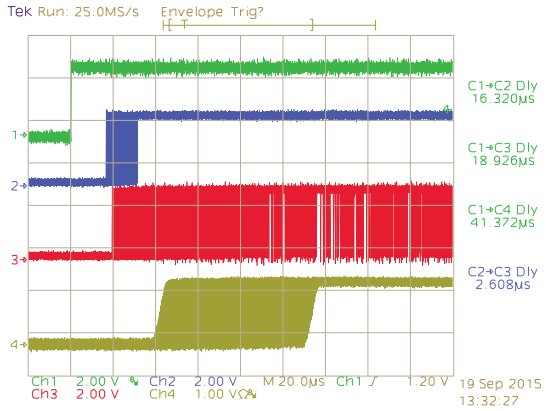


Fig. 11: Accumulated samples of motion-to-photon latency measurement. Same channel labels as Figure 10. Delay times at the right are the minimums of those captured. In this aggregate example, total latency (C1→C4) ranged between 41.372 µs and about 115 µs.

rate) and is equal to about 23 µs; doubling the binary frame rate would approximately halve this figure.

On the right-hand column of Figure 10, several trace-to-trace delays are shown. For this example trace, the value for "C1→C4 Dly," 97.3 µs, is the total latency from the detection of motion (the positive transition of channel 1) to the detection of photons emitted by the display (the positive transition of channel 4).

5.4 Latency Range Analysis

To confirm our characterization of the variable latency sources in the system, we switched the oscilloscope into "envelope" mode, wherein it essentially paints many successive triggered traces on top of each other. An example of this is seen in Figure 11; the positions and colors of the traces are the same as in the preceding discussion. Recalling that the time base is set to 20 µs per division, we can clearly see the fixed and variable components of the serial transmission delay: the minimum distance between the green and blue traces is about 15 µs and, over the many samples taken, the data arrival occurs within about a 15 µs window thereafter. This confirms our analysis of this latency component. Looking at channel 4 (yellow trace), we see the range of times when light was detected. Measuring from the 50% (vertical) point on the left- and right-most edges, we see that the range is consistent with the claimed 70us bounds. Putting together all of the latency data we find the results presented in Table 2.

In summary, we have carefully instrumented and characterized the

Table 2: Summary of Latency Components

Component	*Best*	*Worst*	*Average*
Serial Transmission (C1→C2)	15 µs	2×15 µs	1.5×15 µs
Binary Frame Phase (C2→C3)	0 µs	70 µs	0.5×70 µs
Mirror Refresh (C3→C4)	23 µs	23 µs	23 µs
Motion-to-Photon (C1→C4)	38 µs	123 µs	80.5 µs

sources of latency between physical motion and photon-to-eye delivery within our system. Disregarding nanosecond-scale processing times and speed-of-light delays, all of the latencies are due to fundamental limitations of the present display device and the present interconnect between our tracking sensor and the display processor. Our system can consume and make use of tracking data at rates of tens of kHz. From a technology standpoint, the bottleneck in extending this system is in acquiring and coupling tracking data at least an order of magnitude faster than today's technology.

6 Qualitative Results

We conducted an experiment to assess the efficacy of our implementation. More specifically, our aim is to assess how well our implementation maintains registration between real and augmented objects as the user turns his or her head.

6.1 Experimental Setup

A checkerboard target, with 4 cm squares, is placed within the field of view of the HMD at a distance of about 1.7 m. A corresponding two-color (white and dark gray) virtual checkerboard is displayed on the HMD. As the HMD pans, the two should stay locked together. Due to the nature of the checkerboard pattern, any misregistration will be quite obvious.

A GoPro® Hero 4 Black camera was mounted in the HMD approximately at the location of a user's left eye. The camera is rigidly-attached to the HMD rig. The full assembly is shown in Figure 4.

For the experiment, the HMD is rotated back-and-forth over a range of approximately 30 degrees, which is approximately the horizontal field of view of the display. Rotation was performed by hand. To provide consistent angular velocity among trials, we mounted a protractor scale to the top of the HMD and used a metronome to time the back-and-forth movements. The HMD was rotated such that it moved 10 degrees per beat (for three beats) and then rested for one beat. We executed experiments with metronome speeds of 100 and 300 beats per minute. Thus we see a consistent acceleration, constant velocity, and deceleration in both directions. The resulting angular velocities equate to approximately 17 °/s and 50 °/s, respectively.

Videos were recorded at 240 Hz both with and without our latency correction system enabled at each of the two angular velocities, with clips presented in the accompanying video[4] and some sample frames shown in Figure 1. Our movement technique allowed for nearly-synchronized side-by-side comparisons of the respective tests. While the camera uses a rolling shutter, the horizontal axis of the image plane and the tested motion patterns were aligned, so comparisons along that axis remain valid. We also ran a test using a full frame image with a cutout registered to the same physical checkerboard (see Figure 12).

6.2 Discussion

Representative samples from the resulting videos under fast (50 °/s) motion are shown in Figure 1. Figure 1(a) depicts typical behavior with our latency compensation system disabled. Specifically, the image displayed on the HMD is updated only as new frames arrive from the GPU (at 60 Hz). In that photo, we see that the displayed image lags the real-world image by almost three squares. In general, the augmented image lags behind the real-world image by an amount proportional to the rate of movement and the two worlds come into registration only *after* the motion ceases. It is worth noting that even doubling

[4] http://youtu.be/DvMwDfyjk1E

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2016.2518038, IEEE Transactions on Visualization and Computer Graphics

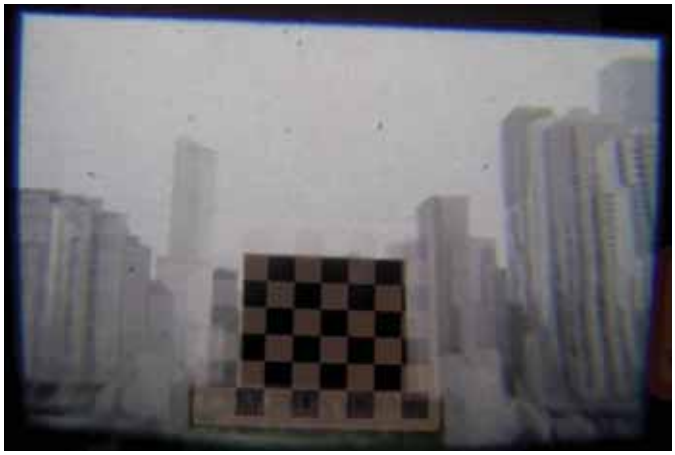


Fig. 12: A sample frame using our latency compensation algorithm under fast panning motion (50 °/s) where the majority of the field of view is dominated by a off-kilter grayscale planar image of a city skyline, in which a small cutout region is registered to the physical checkerboard. Note that the cutout and the checkerboard interior are well aligned.

the GPU frame rate to 120 Hz would not meaningfully mitigate this effect, as the lag would be reduced by at most half the displacement.

The photo in Figure 1(b) depicts the same scenario but with our latency correction system enabled. In this case, the real-time tracking data from the rotary encoders is used to perform our 2-D offset correction, thereby correcting for misregistration due to motion. We observe that the augmented image is well-registered to the real-world object. Under the fast (50 °/s) pan, we also note that the motion blur in the augmented image is consistent with that of the real-world object. The blur is smooth and natural, significantly due to updating the input to our modulation scheme at a high rate; as the tracking information has the potential to update every binary frame, the window on the desired image can also update. The eye or camera continuously integrates these moving binary frames, resulting in motion blur.

In our accompanying video, which presents both slow and fast motion, the virtual checkerboard could occasionally become misaligned to its physical counterpart; this typically happened at one end of the motion path. It is worth noting that at those faulty HMD poses, both the conventional display algorithm and our latency compensating algorithm would agree on the amount of misalignment if the HMD were not moving at the time. The source of this unfortunate misalignment was that the rig to which our HMD was mounted allowed for slight freedom of motion along axes that were not tracked, which repeatedly affected the calibration and caused the static divergence between the real and virtual worlds. This highlights the necessity for using high quality and accurate tracking and calibration schemes in order to support AR applications in which the two worlds must be rigidly aligned. It is worth noting that while actively moving, even in the poor calibration zones, our algorithm kept the two worlds closer.

7 Future Work

In this work, we have presented a system with very low display latency, and because we use a mechanical, zero-latency tracking system for testing, our end-to-end, motion-to-photon latency is also very low. A more general use case would introduce additional challenges: device inertia and low latency general pose tracking. Despite these challenges, the basic design of our display system would continue to benefit these potential systems.

7.1 Inertia

In our development system, we use several commercial development kits (namely the TI DLP® Discovery™ 4100 Kit for projection and the HTG-777 FPGA board for processing) and some custom components (a custom interconnect board and conversion optics). The use of these kits adds both constraints and bulk to the design; mostly these resulted from a lack of high-speed serial links on the DMD chip and controller board used, requiring many very wide parallel links instead, which needed large inflexible cables. Also, the original optics were designed for wall rather than near eye projection; as a result, the light source is much too bright—requiring a neutral density filter—and additional optics are needed to refocus the image to a much closer target. This resulted in a heavy head-mounted display of over 3 kg.

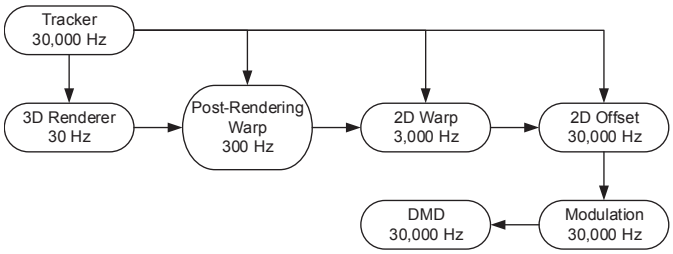


Fig. 13: Theoretical staged render cascade to handle general, unrestricted motion. Rates and ratios between stages are presented for illustrative purposes only.

However, if one were to design a custom DMD-based HMD using our technique from scratch, then several options to reduce the weight (and also the inertia) would be available. Texas Instruments is already designing new, small, low-power DMDs for head-worn displays [6]. Using a smaller light source and a waveguide for optics would also be less bulky. These and specialized (rather than FPGA) control chips with high-speed serial links (at least 13 Gbit/s for the current resolution and binary frame rate) would allow one to further minimize the amount of hardware that needs to go on the head, and move the display processing logic into a backpack or off-person location. Additionally, for mass production, the processing hardware could be optimized further for minimal size and power or produced as ASICs rather than general-purpose (reprogrammable) FPGAs.

7.2 Latency and General Pose Tracking

Tracking accuracy and latency from more general purpose trackers represent a more significant problem for OST AR displays. Our technique is an open-loop process and only performs dynamic correction due to latency from initial rendering to scanout to presentation. Errors due to miscalibration or late reports directly from the tracking system to the display processor would both degrade the experience by misaligning the virtual imagery or causing it to lag behind the physical world. However, in the presence of a tracking system which reports frequently, though with significant delay, we could continue to reduce the perceived latency to the sum of that tracking latency and about 80 µs of display latency, which would be less than a standard display system where the latency also includes GPU processing, buffering, and transmission and display buffering, processing, and outputting. Tracking systems with low update rates, such as those using cameras, would reduce the effectiveness of the display, as the virtual imagery only moves with new tracker poses. AR systems using tracking systems with low latency and high report rates would gain the most benefit from our display processing system.

Assuming that a sufficiently high-frequency, low latency, unrestricted, six degree-of-freedom (x, y, z, roll, pitch, yaw) tracking system existed, then some additional modifications to the display process would be useful to further improve the quality of the latency correction. Our current display algorithm takes advantage of the alignment of the motion rotation axes and the DMD's pixels; it allows us to perform raster order reads of windows of a source image for modulation processing using only a global 2D translation offset. Performing more general warps including rotation, skew, perspective, homography, or distortion would significantly complicate the processing. However, it could be staged in a sequence of post-rendering (and post-transmission) warps, where intermediate stages perform the best possible correction at the best possible speed each using the prior stage's

processing-time tracker pose and the live tracker pose. While each stage would add a bit more processing time, each later stage would perform a smaller correction to account for the overall latency, so the assumption of small pose changes would continue to hold. A sample render cascade pipeline is shown in Figure 13.

8 Conclusion

We have introduced a novel image generation pipeline capable of multi-kilohertz update rates with an average total latency of 80 µs. The system uses a conventional GPU for image generation, coupled with kilohertz-rate just-in-time pitch-yaw corrections executed in an FPGA-based controller that drives a Digital Micromirror Device. The final imagery consists of binary frames updated at over 15 kHz. Greyscale (and in the future also color) imagery is achieved through a novel, low-overhead pseudorandom pulse density modulation technique yielding a perceived image quality that approaches the one produced by the theoretical ideal (delta-sigma), at much lower cost and thus suitable for low-power implementations. The accompanying video demonstrates our fully operational lab prototype and its performance in an Augmented Reality scenario.

As latency is the dominant source of registration errors in Augmented Reality, as well as the dominant cause of user discomfort in Virtual Reality, this technology and its future variants will likely become essential components of widely used VR and AR systems.

Other future work in this area should include accurate minimal-latency tracking at or above 10 kHz, improved system calibration, just-in-time kilohertz-rate correction for additional degrees of freedom such as head roll, and of course, adaptation to other display technologies suitable for mobile devices (e.g. AMOLED).

Acknowledgments

The authors wish to thank Kurtis Keller and Jim Mahaney for their engineering advice and support. They also wish to thank Mary Whitton for her insights and suggestions. This research was supported in part by NSF grant CHS IIS-1423059: Minimal-Latency Tracking and Display for Head-Worn Augmented Reality Systems. This research was also supported in part by the BeingThere Centre, a collaboration between Eidgenössische Technische Hochschule (ETH) Zürich, Nanyang Technological University (NTU) Singapore, and University of North Carolina (UNC) at Chapel Hill. The Centre is supported by these three institutions and by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media Programme Office.

References

[1] AMD. AMD FreeSync™ technology. http://www.amd.com/en-us/innovations/software-technologies/technologies-gaming/freesync, 2015.

[2] P. Aziz, H. Sorensen, and J. van der Spiegel. An overview of sigma-delta converters. *Signal Processing Magazine, IEEE*, 13(1):61–84, Jan 1996.

[3] R. T. Azuma. *Predictive Tracking for Augmented Reality*. PhD thesis, University of North Carolina at Chapel Hill, 1995.

[4] R. T. Azuma. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.

[5] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *Computer Graphics and Applications, IEEE*, 15(5):52–60, 1995.

[6] V. R. Bhakta, J. Richuso, and A. Jain. DLP® technology for near eye display. White Paper DLPA051, Texas Instruments, September 2014.

[7] G. Bishop, H. Fuchs, L. McMillan, and E. J. S. Zagier. Frameless rendering: Double buffering considered harmful. In *Proc. of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 175–176, New York, NY, USA, 1994. ACM.

[8] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[9] 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, 2012.

[10] A. Dayal, C. Woolley, B. Watson, and D. Luebke. Adaptive frameless rendering. In *Proc. Eurographics Conference on Rendering Techniques*, pages 265–275, 2005.

[11] R. L. Holloway. Registration Error Analysis for Augmented Reality. *Presence*, 6(4):413–432, 1997.

[12] L. J. Hornbeck. Digital light processing™ for high brightness, high-resolution applications. In *Proc. SPIE, Projection Displays III*, pages 27–40, 1997.

[13] L. J. Hornbeck. A digital driving technique for an 8b QVGA AMOLED display using ΔΣ modulation. In *Proc. IEEE International Solid State Circuits Conference*, pages 270–272, 2009.

[14] M. C. Jacobs, M. A. Livingston, and A. State. Managing latency in complex augmented reality systems. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, I3D '97, pages 49–ff., New York, NY, USA, 1997. ACM.

[15] J. J. Jerald. *Scene-Motion- and Latency-Perception Thresholds for Head-Mounted Displays*. PhD thesis, University of North Carolina at Chapel Hill, 2009.

[16] J. T. Kajiya and J. Torborg. Talisman: Commodity realtime 3d graphics for the PC. Association for Computing Machinery, Inc., January 1996.

[17] J. Koeter. Whats an LFSR. Technical Report SCTA036A, Texas Instruments, December 1996.

[18] M. Livingston and Z. Ai. The effect of registration error on tracking distant augmented objects. In *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 77–86, Sept 2008.

[19] W. R. Mark, L. McMillan, and G. Bishop. Post-Rendering 3D Warping. In *Proc. ACM I3D*, pages 7–16, 1997.

[20] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 39–46, New York, NY, USA, 1995. ACM.

[21] M. Mine and G. Bishop. Just-in-time pixels. Technical report, Chapel Hill, NC, USA, 1995.

[22] NVIDIA Corporation. G-SYNC – Technology – GeForce. http://www.geforce.com/hardware/technology/g-sync/technology, October 2013.

[23] M. Olano, J. D. Cohen, M. R. Mine, and G. Bishop. Combatting rendering latency. In *Proc. ACM I3D*, pages 19–24, 1995.

[24] V. Popescu, J. Eyles, A. Lastra, J. Steinhurst, N. England, and L. Nyland. The WarpEngine: An architecture for the post-polygonal age. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 433–442, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[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, NY, USA, 1994. ACM.

[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, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[27] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 231–242, New York, NY, USA, 1998. ACM.

[28] F. Smit, R. van Liere, and B. Froehlich. A programmable display layer for virtual reality system architectures. *Visualization and Computer Graphics, IEEE Transactions on*, 16(1):28–42, Jan 2010.

[29] F. A. Smit, R. van Liere, and B. Fröhlich. An image-warping VR-architecture: Design, implementation and applications. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, pages 115–122, New York, NY, USA, 2008. ACM.

[30] B. Walter, G. Drettakis, and S. Parker. Interactive rendering using the render cache. In D. Lischinski and G. Larson, editors, *Rendering techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering)*, volume 10, pages 235–246, New York, NY, Jun 1999. Springer-Verlag/Wien.

[31] F. Zheng, T. Whitted, A. Lastra, P. Lincoln, A. State, A. Maimone, and H. Fuchs. Minimizing latency for augmented reality displays: Frames considered harmful. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 195–200, Sept 2014.