Temporal Resolution Multiplexing: Exploiting the limitations of spatio-temporal vision for more efficient VR rendering

Gyorgy Denes* Kuba Maruszczyk† George Ash‡ Rafał K. Mantiuk§

University of Cambridge, UK

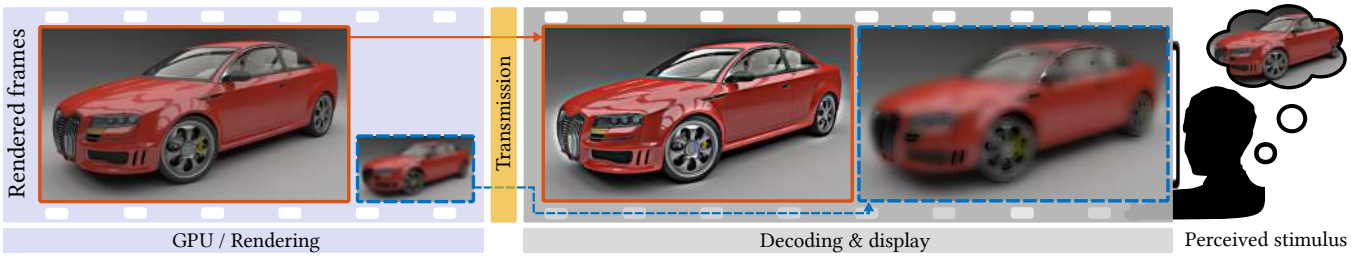

Figure 1: Our technique renders every second frame at a lower resolution to save on rendering time and data transmission bandwidth. Before the frames are displayed, the low resolution frames are upsampled and high-resolution frames are compensated for the lost information. When such a sequence is viewed at a high frame rate, the frames are perceived as though they were rendered at full resolution.

Abstract

Rendering in virtual reality (VR) requires substantial computational power to generate 90 frames per second at high resolution with good-quality antialiasing. The video data sent to a VR headset requires high bandwidth, achievable only on dedicated links. In this paper we explain how rendering requirements and transmission bandwidth can be reduced using a conceptually simple technique that integrates well with existing rendering pipelines. Every even-numbered frame is rendered at a lower resolution, and every odd-numbered frame is kept at high resolution but is modified in order to compensate for the previous loss of high spatial frequencies. When the frames are seen at a high frame rate, they are fused and perceived as high resolution and high-frame-rate animation. The technique relies on the limited ability of the visual system to perceive high spatio-temporal frequencies. Despite its conceptual simplicity, correct execution of the technique requires a number of non-trivial steps: display photometric temporal response must be modeled, flicker and motion artifacts must be avoided, and the generated signal must not exceed the dynamic range of the display. Our experiments, performed on a high-frame-rate LCD monitor and OLED-based VR headsets, explore the parameter space of the proposed technique and demonstrate that its perceived quality is indistinguishable from full-resolution rendering. The technique is an attractive alternative to resolution reduction for all frames, which is a current practice in VR rendering.

Keywords: Temporal multiplexing, rendering, graphics, perception, virtual reality

*e-mail: gyorgy.denes@cl.cam.ac.uk
†e-mail: kuba.maruszczyk@cl.cam.ac.uk
‡e-mail: ga354@cl.cam.ac.uk
§e-mail: rafal.mantiuk@cl.cam.ac.uk

1 Introduction

Increasingly higher display resolutions and refresh rates often make real-time rendering prohibitively expensive. In particular, modern VR systems are required to render binocular stereo views at high frame rates (90 Hz) with minimum latency so that the generated views are perfectly synchronized with head motion. Since current generation VR displays offer a low angular resolution of about 10 pixels per visual degree, each frame needs to be rendered with strong anti-aliasing. All these requirements result in excessive rendering cost, which can only be met by power-hungry, expensive graphics hardware.

The increased resolution and frame rate also pose a challenge for transmitting frames from the GPU to the display. For this reason, VR headsets require high-bandwidth wireless links or cables. When we consider 8K resolution video, even transmission over a cable is problematic and requires compression.

We propose a technique for reducing both bandwidth and rendering cost for high-frame-rate displays by 37–49% with only marginal computational overhead and small impact on image quality. Our technique, Temporal Resolution Multiplexing (TRM), does not only address the renaissance of VR, but can be also applied to future high-refresh-rate desktop displays and television sets to improve motion quality without significantly increasing the bandwidth required to transmit each frame.

TRM takes advantage of the limitations of the human visual system: the finite integration time that results in fusion of rapid temporal changes, along with the inability to perceive high spatio-temporal frequency signals. An illusion of smooth high-frame-rate motion is generated by rendering a low-resolution version of the content for every odd frame, compensating for the loss of information by modifying every even frame. When the even and odd frames are viewed at high frame rates ($> 90 Hz$), the visual system fuses them and perceives the original, full resolution video. The proposed technique, although conceptually simple, requires much attention to details such as display calibration, overcoming dynamic range limitations, ensuring that potential flicker is invisible, and designing a solution that will save both rendering time and bandwidth. We also explore the effect of the resolution reduction factor on perceived quality, and thoroughly validate the method on a high-

frame-rate LCD monitor and two different VR headsets with OLED displays. Our method is simple to integrate into existing rendering pipelines, fast to compute, and can be combined with other common visual coding methods such as chroma-subsampling and video codecs, such as JPEG XS, to further reduce bandwidth.

The main contributions of this paper are:

- A method for rendering and visual coding high-frame-rate video, which can substantially reduce rendering and transmission costs;
- Analysis of the method in the context of display technologies and visual system limitations;
- A series of experiments exploring the strengths and limitations of the method.

2 Related work

Temporal multiplexing, taking advantage of the finite integration time of the visual system, has been used for improving *display resolution* for moving images [10], projectors [29, 15], and for wobulating displays [1, 4]. Temporal multiplexing has been also used to *increase perceived bit-depth* (spatio-temporal dithering) [22] and *color gamut* [17]. It is widely used in digital projectors combining a color wheel with a white light source to produce color images.

The proposed method employs temporal multiplexing to reduce rendering cost and transmission bandwidth for pixel data, which are both major bottlenecks in VR. In this section, we review the most relevant methods that share similar goals with our technique.

2.1 Temporal coherence in rendering

Since consecutive frames in an animation sequence tend to be similar, exploiting the temporal coherence is an obvious direction for reducing rendering cost. A comprehensive review of temporal coherence techniques can be found in [30]. Here, we focus on the methods that are the most relevant for our target VR application: reverse and forward reprojection techniques.

The rendering cost can be significantly reduced if only every k-th frame is rendered, and in-between frames are generated by transforming the previous frame. *Reverse reprojection* techniques [23] attempt to find a pixel in the previous frame for each pixel in the current frame. This requires finding a reprojection operator mapping pixel screen coordinates from the current to the previous frame and then testing whether the current point was visible in the previous frame. Visibility can be tested by comparing depths for the current and previous frames. *Forward reprojection* techniques map every pixel in the previous frame to a new location in the current frame. Such a scattering operation is not well supported by graphics hardware, making a fast implementation of forward reprojection more difficult. This issue, however, can be avoided by warping the previous frame into the current frame [11]. This warping involves approximating motion flow with a coarse mesh grid and then rendering the forward-reprojected mesh grid into a new frame. Since parts of the warped mesh can overlap the other parts, both spatial position and depth need to be reprojected and the warped frame needs to be rendered with depth testing. We discuss the technique of Didyk et al. [11] in more detail in Section 6 as it exploits similar limitations of the visual system as our method.

Commercial VR rendering systems use reprojection techniques to avoid skipped and repeated frames when the rendering budget is exceeded. These techniques may involve rotational forward reprojection [33], which is sometimes combined with screen-space warping, such as asynchronous spacewarp (ASW) [2]. Rotational reprojection assumes that the positions of left- and right-eye virtual cameras are unchanged and only view direction is altered. This assumption is incorrect for actual head motion in VR viewing as the position of both eyes changes when the head rotates. More advanced positional reprojection techniques are considered either too expensive or are likely to result in color bleeding with multi-sample anti-aliasing, introduce difficulty in handling translucent surfaces and dynamic light conditions, and require hole fillings for occluded pixels. Reprojection techniques are considered a last-resort option in VR rendering, used only to avoid skipped or repeated frames. When the rendering budget cannot be met, lowering the frame resolution is preferred over reprojection [33]. Another limitation of reprojection techniques is that there is no bandwidth reduction when transmitting pixels from the GPU to a VR display.

2.2 High-frame-rate display technologies

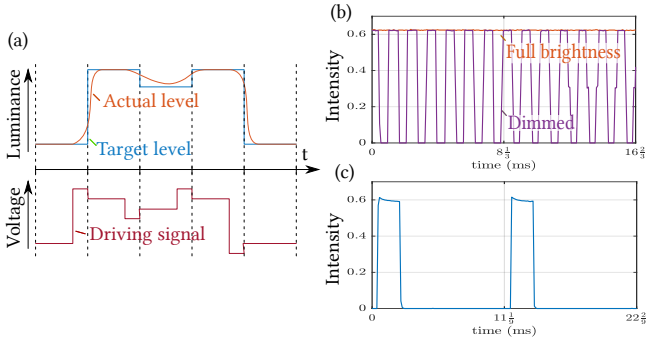

Figure 2: (a) Delayed response of an LCD display driving with a signal with *overdrive*. The plot is for illustrative purposes and it does not represent measurements. (b) Measurement of an LCD (Dell Inspiron 17R 7720) at full brightness and when dimmed, showing all white pixels in both cases. (c) Measurement of HTC Vive display showing all white pixels. Measurements taken with a 9 kHz irradiance sensor.

In this section we discuss issues related to displaying and viewing high-frame-rate animation using two dominant display technologies: LCD and OLED. The main types of artifacts arising from motion shown on a display can be divided into (1) non-smooth motion, (2) false multiple edges (ghosting), (3) spatial blur of moving regions and (4) flickering. The visibility of such artifacts increases for reduced frame rate, increased luminance, higher speed of motion, increased contrast and lower spatial frequencies [7]. Our technique is designed to avoid all four types of artifacts while reducing the computational and bandwidth requirements of high frame rates.

The liquid crystals in the recent generation of LCD panels have relatively short response times and offer between 160 and 240 frames a second. However, liquid crystals still require time to switch from one state to another, and the desired target state is often not reached within the time allocated for a single frame. This problem is partially alleviated by over-driving (applying higher voltage), so that pixels achieve the desired state faster, as illustrated in Figure 2-(a). Switching from one grey-level to another is usually slower than switching from black-to-white or white-to-black. Such non-linear temporal behavior adds significant complexity to modeling display response, which we will address in Section 4.4.

Response time accounts only for a small amount of the blur visible on LCD screens. Most of the blur is attributed to eye motion over an image that remains static for the duration of a frame [12]. When the eye follows a moving object, the gaze smoothly moves over pixels that do not change over the duration of the frame. This introduces blur in the image that is integrated on the retina, an effect known as *hold-type blur* (refer to Figure 12 for the illustration of this effect). Hold-type blur can be reduced by shortening the time pixels are switched on, either by flashing the backlight [12], or inserting black frames (BFI). Both solutions, however, reduce the peak luminance of the display and may result in visible flicker.

OLED displays offer almost instantaneous response but they still suffer from hold-type blur. Hence, most VR systems employ a low-persistence mode in which pixels are switched on for only a small portion of a frame. In Figure 2-(c) we show the measurements of the temporal response we collected for the HTC Vive headset, which shows that the display remains black for 80% of a frame.

Nonlinearity compensated smooth frame insertion (NCSFI) attempts to reduce hold-on motion blur while maintaining peak luminance [6]. The core algorithm is based on similar principles as our method, as it relies on the eye fusing a blurred and sharpened image pair. However, NCSFI is designed for 50–60 Hz TV content and, as we demonstrate in Section 8, produces ghosting artifacts for high angular velocities typical of user-controlled head motion in VR.

In this work we do not consider displays based on digital micromirror devices, which can offer very fast switching times and therefore are used in ultra-low-latency AR displays [21].

2.3 Coding and transmission

Attempts have been made in the past to blur in-between frames to improve coding performance [13]. These methods rely on the visual illusion of motion sharpening which makes moving objects appear sharper than they physically are. However, no such technique has been incorporated into a coding standard. One issue is that at low velocities motion sharpening is not strong enough, leading to a loss of sharpness, as we discuss in more detail in the next section. In contrast to those methods, our technique actively compensates for the loss of high frequencies and preserves original sharpness for both stationary and moving objects.

VR applications require low-latency and low-complexity coding that can reduce the bandwidth of frames sent from a GPU to a display. Such requirements are addressed in the recent JPEG XS standard (ISO/IEC 21122) [9]. In Section 7.1 we demonstrate how the efficiency of JPEG XS can be further improved when combined with the proposed method.

3 Perception of High-Frame-Rate Video

To justify our approach, we first discuss the visual phenomena and models that our algorithm relies on. Most artificial light sources, including displays, flicker with a very high frequency – so high that we no longer see flicker, but rather an impression of steady light. Displays with LED light sources control their brightness by switching the source of illumination on and off at a very high frequency, a practice known as pulse-width-modulation (see Figure 2-(b)). The perceived brightness of such a flickering display will match the brightness of the steady light that has the same time-average luminance — a phenomenon known as the Talbot-Plateau law.

The frequency required for a flickering stimulus to be perceived as steady light is known as the critical fusion frequency (CFF). This frequency depends on multiple factors; it is known to increase proportionally with the log-luminance of a stimulus (Ferry-Porter law), increase with the size of the flickering stimulus, and to be more visible in the parafovea, in the region between 5-30 degrees from the fovea [14].

CFF is typically defined for periodic stimuli with full-on, full-off cycles. With our technique, as the temporal modulation has much lower contrast, flicker visibility is better predicted by the temporal sensitivity [34] or the spatio-temporal contrast sensitivity functions (stCSF) [19]. Such sensitivity models are defined as functions of spatial frequency, temporal frequency and background luminance, where the dimensions are not independent [8]. The visibility of moving objects is better predicted by the spatio-velocity contrast sensitivity function (svCSF) [18], where temporal frequency is replaced with retinal velocity in degrees per second. The contour plots of stCSF and svCSF are shown in Figure 3. The stCSF plot on the left shows that the contours of equal sensitivity form almost straight lines for high temporal and spatial frequencies, suggesting

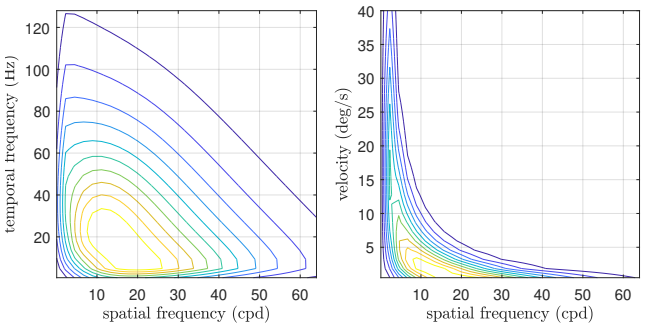

Figure 3: Contour plots of spatio-temporal contrast sensitivity (left) and spatio-velocity contrast sensitivity (right). Based on Kelly's model [18]. Different line colors represent individual levels of relative sensitivity from low (purple/dark lines) to high (yellow/bright lines).

that the sensitivity can be approximated by a plane. This observation, captured in the window of visibility [35] and the pyramid of visibility [34], offer simplified models of spatio-temporal vision, featuring an insightful analysis of visual system limitations in the Fourier domain that we rely on in Section 6.

Temporal vision needs to be considered in conjunction with eye motion. When fixating, the eye drifts around the point of fixation (0.8–0.15 $^{deg}/_{s}$). When observing a moving object, our eyes attempt to track it with speeds of up to 100 $^{deg}/_{s}$, thus stabilizing the image of the object on the retina. Such tracking, known as smooth pursuit eye motion (SPEM) [28], is not perfect, the eye tends to lag behind an object, moving approximately 5-20% slower [8]. However, no drop in sensitivity was observed for velocities up to 7.5 $^{deg}/_{s}$ [20] and only a moderate drop of perceived sharpness was reported for velocities up to 35 $^{deg}/_{s}$ [36]. Blurred images appeared sharper when moving with speeds above 6 $^{deg}/_{s}$ and the perceived sharpness of blurred images was close to that of sharp moving images for velocities above 35 $^{deg}/_{s}$ [36]. This effect, known as *motion sharpening*, can aid us to see sharp objects when retinal images are blurry because of imperfect SPEM tracking by the eye. Motion sharpening is also attributed to a well-known phenomenon where video appears sharper than individual frames. Takeuchi and De Valois demonstrated that this effect corresponds to the increase of luminance contrast in medium and high spatial frequencies [31]. They also demonstrated that interleaved blurry and original frames can appear close to the original frames as long as the cut-off frequency of the low-pass filter is sufficiently large. Our method benefits from motion sharpening, but it cannot fully rely on it as the sharpening is too weak for low velocities.

4 Temporal Resolution Multiplexing

Our main goal is to reduce both the bandwidth and computation required to drive high-frame-rate displays (HFR), such as those used in VR headsets. This is achieved with a simple, yet efficient algorithm that leverages the eye's much lower sensitivity to signals with both high spatial and temporal frequencies.

Our algorithm, Temporal Resolution Multiplexing (TRM), operates on reduced-resolution render targets for every even-numbered frame – reducing both the number of pixels rendered and the amount of data transferred to the display. TRM then compensates for the contrast loss, making the reduction almost imperceivable.

The diagram of our processing pipeline is shown in Figure 4. We consider *rendering & encoding* to be a separate stage from *decoding & display* as they may be realized in different hardware devices: typically rendering is performed by a GPU, and decoding & display is performed by a VR headset. The separation into two parts is designed to reduce the amount of data sent to a display. The optional

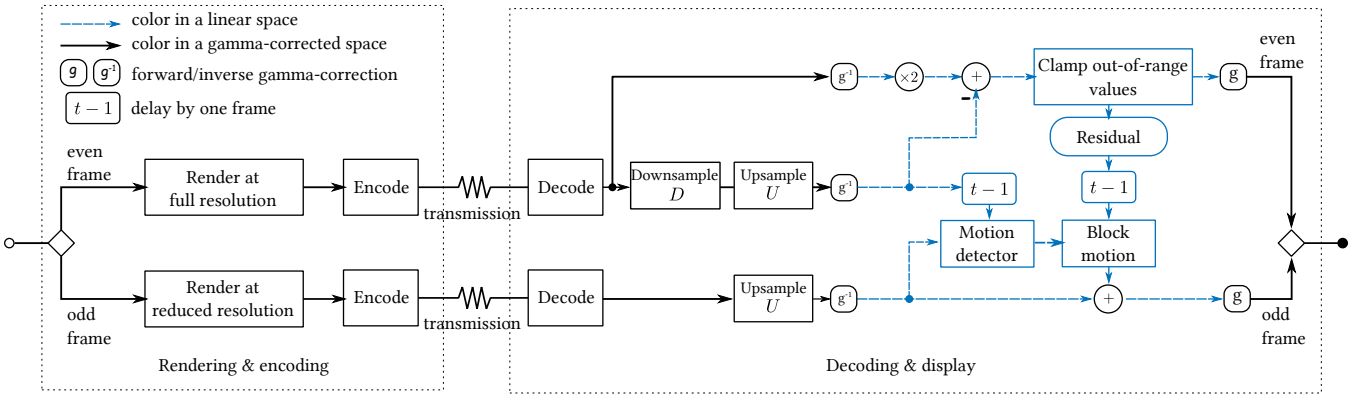

Figure 4: The processing diagram for our method. Full- and reduced-resolution frames are rendered sequentially, thus reducing rendering time and bandwidth for reduced resolution frames. Both types of frames are processed so that when they are displayed in rapid succession, they appear the same as the full resolution frames.

encoding and decoding steps may involve chroma sub-sampling, entropy coding or a complete high-efficiency video codec, such as h265 or JPEG XS. All of these bandwidth savings would come on top of a 37–49% reduction from our method.

The top part of Figure 4 illustrates the pipeline for even-numbered frames, rendered at full resolution, and the bottom part the pipeline for odd-numbered frames, rendered at reduced resolution. The algorithm transforms those frames to ensure that when seen on a display, they are perceived to be almost identical to the full-resolution and full-frame-rate video. In the next sections we justify why the method works (Section 4.1), explain how to overcome display dynamic range limitations (Section 4.2), address the problem of phase distortions (Section 4.3), and ensure that we can accurately model light emitted from the display (Section 4.4).

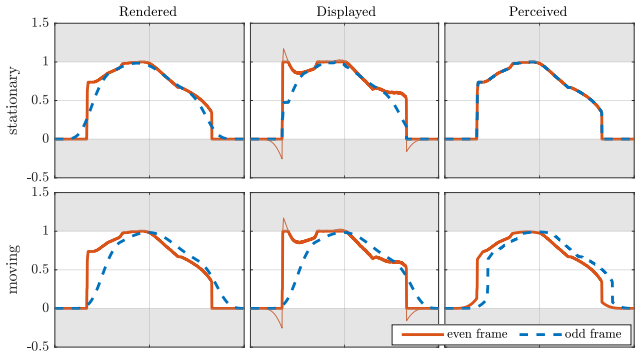

Figure 5: Illustration of the TRM pipeline for stationary (top) and moving (bottom) objects. The two line colors denote odd- and even-numbered frames. After *rendering*, the full-resolution even-numbered frame (continuous orange) needs to be sharpened to maintain high-frequency information. Values lost due to clamping are added to the low-resolution frame (dashed blue), but only whenever the object is not in motion, i.e. *displayed* stationary low-resolution frames are different from the *rendering*, whereas moving ones are identical. Consequently, stationary objects are always perfectly recovered, while moving objects may lose a portion of high-frequency details.

4.1 Frame integration

We consider our method suitable for frame rates of 90Hz or higher, with frame duration 11.1 ms or less. A pair of such frames lasts approx. 22.2 ms, which is short enough to fit within the range in which the Talbot-Plateau law holds. Consequently, the perceived stimulus is the average of two consecutive frames, one containing mostly low frequencies (reduced resolution) and the other containing all frequencies. Let us denote the upsampled reduced-resolution (odd) frame at time instance t with α_t:

$$\alpha_t(x,y) = (U \circ i_t)(x,y)\ , \quad t = 1,3,... \tag{1}$$

where U is the upsampling operator, i_t is a low-resolution frame and $\circ$ denotes function composition. Upsampling in this context means interpolation and increasing sampling rate. When we refer to downsampling, we mean the application of an appropriate low-pass filter and resolution reduction. Note that i_t must be represented in linear colorimetric values (not gamma compressed). We will consider only luminance here, but the same analysis applies to the red, green and blue color channels. The initial candidate for the all-frequency even frame, compensating for the lower resolution of the odd-numbered frame, will be denoted by β:

$$\beta_t(x,y) = 2I_t(x,y) - (U \circ D \circ I_t)(x,y)\ , \quad t = 2,4... \tag{2}$$

where D is a downsampling function that reduces the size of frame I_t to that of i_t ($i_t = D \circ I_t$), and U is the upsampling function, the same as that used in Equation 1. Note that when an image is static ($I_t = I_{t+1}$), according to the Talbot-Plateau law, the perceived image is:

$$\alpha_t(x,y) + \beta_{t+1}(x,y) = 2I_t(x,y)\ . \tag{3}$$

Therefore, we perceive the image I_t at its full resolution and brightness (the equation is the sum of two frames and hence $2I_t$). A naïve approximation of $\beta_t(x,y) = I_t(x,y)$ would result in a loss of contrast for sharp edges, and images that appears overly *soft*.

The top row in Figure 5 illustrates rendered low- and high-frequency components (1st column), compensation for missing high frequencies (2nd column), and the perceived signal (3rd column), which is identical to the original signal if there is no motion. However, what is more interesting and non-obvious is that we will see a correct image even when there is movement in the scene. If there is movement, it is most likely caused by an object or camera motion. In both cases, the gaze follows an object or scene motion (see SPEM in Section 3), thus fixing the image on the retina. As long as the image is fixed, the eye will see the same object at the same retinal position and Equation 3 will be valid. Therefore, as long as the change is due to rigid motion trackable by SPEM, the perceived image corresponds to the high resolution frame I.

4.2 Overshoots and undershoots

The decomposition into low- and high-resolution frames α and β is not always straightforward as the high resolution frame β may contain values that exceed the dynamic range of a display. As an example, let us consider the signal shown in Figure 5 and assume that our display can reproduce values between 0 and 1. The compensated high-resolution frame β, shown in orange, contains values that are above 1 and below 0, which we refer to as overshoots and undershoots. If we clamp the "orange" signal to the valid range, the perceived integrated image will lose some high-frequency information and will be effectively blurred. In this section we explain how this problem can be reduced to the point that the loss of sharpness is imperceptible.

For stationary pixels, overshoots and undershoots do not pose a significant problem. The difference between an enhanced even-numbered frame β_t (Equation 2) and the actually displayed frame, altered by clamping to the display dynamic range, can be stored in the *residual buffer* ρ_t. The values stored in the residual buffer are then added to the next low resolution frame: $\alpha'_{t+1} = \alpha_{t+1} + \rho_t$. If there is no movement, adding the residual values restores missing high frequencies and reproduces the original image. However, for pixels containing motion, the same approach would introduce highly objectionable ghosting showing as a faint copy of sharp edges at the previous frame locations.

In practice, better animation quality is achieved if the residual is ignored for moving pixels. This introduces a small amount of blur for a rare occurrence of high-contrast moving objects, but such blur is almost imperceptible due to motion sharpening (see Section 3). We therefore apply a weighing mask when adding the residual to the odd-numbered frame.

$$\alpha'_{t+1}(x,y) = \alpha_{t+1} + w(x,y)\,\rho_t(x,y)\,, \tag{4}$$

where $\alpha'(x,y)$ is the final displayed odd-numbered frame. For $w(x,y)$ we first compute the contrast between consecutive frames as an indicator of motion:

$$c(x,y) = \frac{|U \circ D \circ I_{t-1}(x,y) - U \circ i_t(x,y)|}{U \circ D \circ I_{t-1}(x,y) + U \circ i_t(x,y)} \tag{5}$$

then apply a soft-thresholding function:

$$w(x,y) = \exp\left(-s\,c_t(x,y)\right)\,, \tag{6}$$

where s is an adjustable parameter controlling the sensitivity to motion. It should be noted that we avoid potential latency issues in motion detection by computing the residual weighing mask after the rendering of the low-resolution frame.

The visibility of blur for moving objects can be further reduced if we upsample and downsample images in the appropriate color space. Perception of luminance change is strongly non-linear, blur introduced in dark regions tends to be more visible than in bright regions. The visibility of blur can be more evenly distributed between dark and bright pixels if upsampling and downsampling operations are performed in a gamma-compressed space, as shown in Figure 6. A cubic root-function is considered a good predictor of brightness, and is commonly used in uniform color spaces, such as CIE Lab and CIE Luv. However, the standard sRGB colorspace with gamma $\approx$2.2 is sufficiently close to the cubic root ($\gamma = 3$) and, since the rendered or transmitted data is likely to be already in that space, it provides a computationally efficient alternative.

4.3 Phase distortions

A naïve rendering of frames at reduced resolution without anti-aliasing results in a discontinuity of phase changes for moving objects, which reveals itself as juddery motion. A frame that is rendered at lower resolution and upsampled is not equivalent to the same frame rendered at full resolution and low-pass filtered, as it is not only missing information about high spatial frequencies, but also lacks accurate phase information.

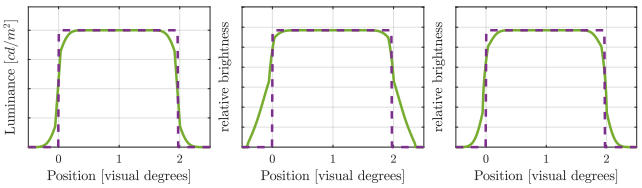

Figure 6: Averaged (solid) vs. original (dashed) frames after our algorithm for moving square-wave signal. *Left*: In linear space over- and undershoot artifacts are equally sized; however, such representation is misleading, as brightness perception is non-linear. *Center*: better estimation of perceived signal using Stevens's brightness, where overshoot artifacts are predicted more noticeable. Right: TRM performs sampling in γ-compressed space, the perceptual impact of over- and undershoot artifacts are balanced (in Steven's brightness).

In practice, the problem can be mitigated by rendering with MSAA. Custom Gaussian, bicubic or Lanczos filters can further improve the results, but should only be used when there is native hardware support [26]. Alternatively, the low-resolution frame can be low-pass filtered to achieve similar results.

In our experiments we used a Gaussian filter with $\sigma = 2.5$ pixels for both the downsampling operator D and for MSAA resolve. Upsampling was performed as bilinear interpolation, as it is fast and supported by GPU texture samplers. Better upsampling operators, such as Lanczos, could be considered in the future.

4.4 Display models

The frame-integration property of the visual system, discussed in Section 4.1, applies to physical quantities of light, but not to gamma-compressed pixel values stored in frame buffers. Small inaccuracies in the estimated display response can lead to over- or under-compensation in high-resolution frames. Therefore, it is essential to accurately characterize the display.

OLED (HTC Vive, Oculus Rift)

OLED displays can be found in consumer VR headsets including the HTC Vive and the Oculus Rift. These can be described accurately using standard parametric display models, such as gain-gamma-offset [3]. However, in our application, gain does not affect the results and offset is close to 0 for near-eye OLED displays. Therefore, we ignore both gain and offset and model the display response as a simple gamma: $I = v^\gamma$, where I is a pixel value in linear space (for an arbitrary color channel), v is the pixel value in gamma-compressed space and γ is a model parameter. In practice, display manufacturers often deviate from the standard $\gamma \approx 2.2$ and the parameter tends to differ between color channels. To avoid chromatic shift, we measured the display response of the HTC Vive and Oculus Rift CV1 with a Specbos 1211 spectroradiometer for full-screen color stimuli (red, green, blue), finding separate γ values for the three primaries. To accommodate high peak luminance levels, each measurement was repeated through a neutral density filter (Kodak gelatine ND 1.0). Measurements were aggregated accounting for measurement noise and the transmission properties of the filter. The best fitting parameters were $\gamma_r = 2.2912$, $\gamma_g = 2.2520$ and $\gamma_b = 2.1940$ for our HTC Vive and $\gamma_r = 2.1526$, $\gamma_g = 2.0910$ and $\gamma_b = 2.0590$ for the Oculus.

HFR LCD (ASUS P279Q)

Due to the finite and different rising and falling response times of liquid crystals discussed in Section 2.2, we need to consider the previous pixel value when modeling the per-pixel response of an

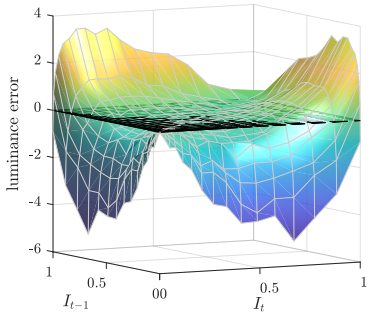

Figure 7: Luminance difference between measured luminance value and expected ideal luminance (sum of two consecutive frames) for alternating I_t and I_{t-1} pixel values. Our measurements for ASUS ROG Swift P279Q indicate a deviation from the plane when one of the pixels is significantly darker or brighter than the other.

LCD. We used a Specbos 1211 with a 1 s integration time to measure alternating pixel value pairs displayed at 120 Hz on an ASUS ROG Swift P279Q. Figure 7 illustrates the difference between predicted luminance values (sum of two linear values, estimated by gain-gamma-offset model) and actual measured values. The inaccuracies are quite substantial, especially for low luminance, resulting in haloing artifacts in the fused animations.

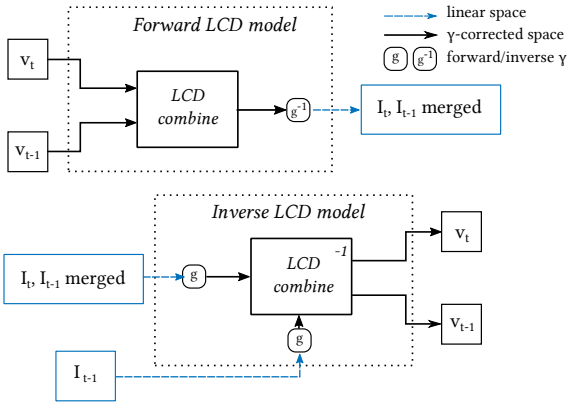

Figure 8: Schematic diagram of our extended LCD display model for high-frame-rate monitors. **a)** In the forward model two consecutive pixel values are combined before applying inverse gamma. **b)** The inverse model applies gamma before inverting the LCD combine step. The previous pixel value is provided to find a $\langle v_t, v_{t-1} \rangle$ pair, where $v_{t-1}^{\gamma} \approx I_{t-1}$

To accurately model LCD response, we extend the display model to account for the pixel value in the previous frame. The forward display model, shown in the top of Figure 8, contains an additional *LCD combine* block that predicts the equivalent gamma-compressed pixel value, given pixel values of the current and previous frames. Such a relation is well-approximated by a symmetric bivariate quadratic function of the form:

$$M(v_t, v_{t-1}) = p_1(v_t^2 + v_{t-1}^2) + p_2 v_t v_{t-1} + p_3(v_t + v_{t-1}) + p_4 \,, \quad (7)$$

where $M(v_t, v_{t-1})$ is the merged pixel value, v_t and v_{t-1} are the current and previous gamma-compressed pixel values and $p_{1..4}$ are the model parameters. To find the inverse display model, the inverse of the merge function needs to be found. The merge function is not strictly invertible as multiple combinations of pixel values can produce the same merged value. However, since we render in real-time and can control only the current but not the previous frame, v_{t-1} is already given and we only need to solve for v_t. If the quadratic equation leads to a non-real solution, or a solution outside the display dynamic range, we clamp v_t to be within 0..1 and then solve for v_{t-1}. Although we cannot fix the previous frame as it has already been shown, we can still add the difference between the desired value and the displayed value to the residual buffer ρ, taking advantage of the correction feature in our processing pipeline. The difference in prediction accuracy for a single-frame and our temporal display model is shown in Figure 9.

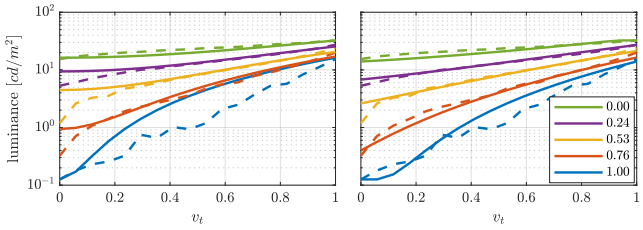

Figure 9: Dashed lines: measured display luminance for red primaries (v_t), given a range of different v_{t-1} pixel values (line colors). Solid lines: predicted values without temporal display model (left) and with our temporal model (right).

5 Experiment 1: Resolution Reduction vs. Frame Rate

To analyze how the display and rendering parameters, such as refresh rate and reduction factor, affect the motion quality of TRM rendering, we conducted a psychophysical experiment. In the experiment we measure the maximum possible resolution reduction factor while maintaining perceptually indistinguishable quality from standard rendering.

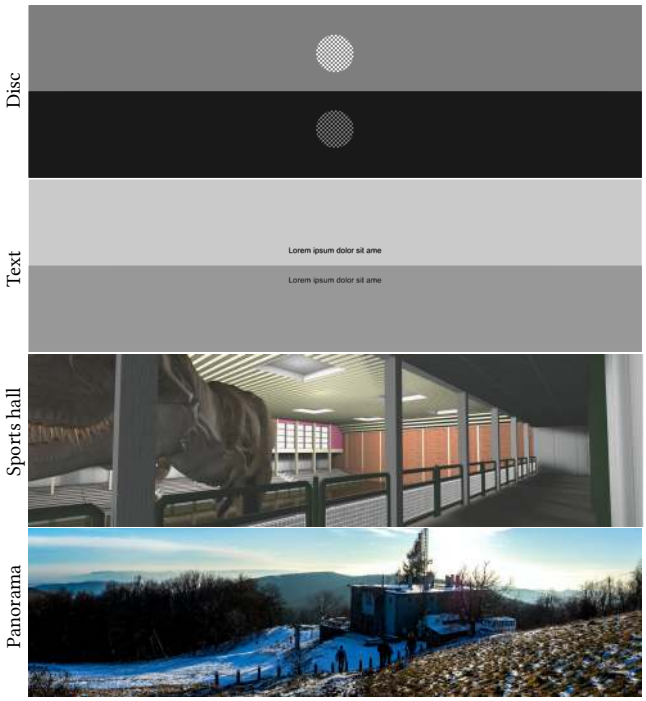

Figure 10: Stimuli used for Experiment 1.

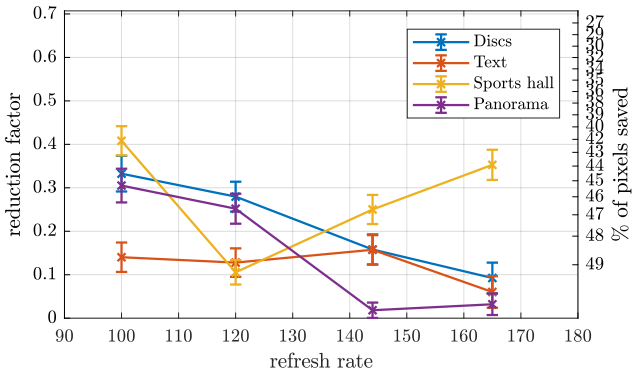

Figure 11: Result of Experiment 1: finding the smallest resolution reduction factor for four scenes and four display refresh rates. As the reduction is applied to both horizontally and vertically, the % of pixels saved over a pair of frames can be computed as $(1-r^2)/2$.

Setup:

The animation sequences were shown on a 2560×1440 (WQHD) high-frame-rate Asus ROG Swift P279Q 27". The display allowed us to finely control the refresh rate, unlike any OLED displays found in VR headsets. The viewing distance was fixed at 75cm using a headrest, resulting in the angular resolution of 56 pixels per degree. Custom OpenGL software was used to render the sequences in real-time, with or without TRM.

Stimuli:

In each trial participants saw two short animation sequences (avg. 6s) one after another, one of them rendered using TRM, the other rendered at the full resolution. Both sequences were shown at the same frame-rate. Figure 10 shows a thumbnail of the four animations used in the experiment. The animations contained moving *Discs*, scrolling *Text*, panning of a *Panorama* and a 3D model of a *Sports hall*. The two first clips were designed to provide an easy-to-follow object with high contrast; the two remaining clips tested the algorithm on rendered and camera-captured scenes. *Sports hall* tested interactive applications by letting users rotate the camera with a mouse. The other sequences were pre-recorded. In the *Panorama* clip we simulated panning as it provided better control over motion speed than video captured with a camera.

The animations were displayed at four frame rates: 100 Hz, 120 Hz, 144 Hz and 165 Hz. We could not test lower frame rates because the display did not natively support 90 Hz, and flicker was visible at lower frame rates.

Task:

The goal of the experiment was to find the threshold reduction factor at which the observers could notice the difference between TRM and standard rendering with 75% probability. An adaptive QUEST procedure, as implemented in Psychophysics Toolbox extensions [5], was used to sample the continuous scale of reduction factors and to fit a psychometric function. The order of trials was randomized so that 16 QUEST procedures were running concurrently to reduce the learning effect. In each trial the participant was asked to select the sequence that presented *better motion quality*. They had an option to re-watch the sequences (in case of lapse of attention), but were discouraged from doing so. Before each session, participants were briefed about their task both verbally and in writing. The briefing explained the motion quality factors (discussed in Section 2.2) and was followed by a short training session, in which the difference between 40 Hz and 120 Hz was demonstrated.

Participants:

Eight paid participants aged 18 – 35 took part in the experiment. All had normal or corrected-to-normal full color vision.

Results:

The results in Figure 11 show a large variation in the reduction factor from one animation to another. This is expected as we did not control motion velocity or contrast in this experiment, while both factors strongly affect motion quality. For all animations, except *Sports hall*, the resolution of odd-numbered frames can be further reduced for higher refresh-rate displays. *Sports hall* was an exception in that participants chose almost the same reduction factor for both the 100 Hz and 165 Hz display. Post-experiment interviews revealed that the observers used the self-controlled motion speed and sharp edges present in this rendered scene to observe slight variation in sharpness. Note that this experiment tested discriminability, which results in a conservative threshold for ensuring same quality. That means that such small variations in sharpness, though noticeable, are unlikely to be objectionable in practical applications.

Overall, the experiment showed that a reduction factor of 0.4 or less produces animation that is indistinguishable from rendering frames at the full-resolution. Stronger reduction could be possible for high-refresh displays, however, the savings become negligible as the factor is reduced below 0.4.

6 Comparison with other techniques

In this section we compare our technique to other methods intended for improving motion quality or reducing image transmission bandwidth.

Table 1 provides a list of common techniques that could be used to achieve similar goals as our method. The simplest way to halve the transmission bandwidth is to *halve the frame rate*. This obviously results in non-smooth motion and severe hold-type blur. *Interlacing* (odd and even rows are transmitted in consecutive frames) provides a better way to reduce bandwidth. Setting missing rows to 0 can reduce motion blur. Unfortunately, this will reduce peak luminance by 50% and may result in visible flicker, aliasing and combing artifacts. Hold-type blur can be reduced by inserting a black frame every other frame (*black frame insertion — BFI*), or backlight flashing [12]. This technique, however, is prone to causing severe flicker and also reduces peak display luminance. Nonlinearity compensated smooth frame insertion (*NCSFI*) [6] relies on a similar principle as our technique and displays sharpened and blurred frames. The difference is that every pair of blurred and sharpened frames is generated from a single frame (from 60 Hz content). The method saves 50% on computation and does not suffer from reduced peak brightness, but results in ghosting at higher speeds, as we demonstrate in Section 8.

Didyk et al. [11] demonstrated that up to two frames could be morphed from a previously rendered frame. They approximate scene deformation with a coarse grid that is snapped to the geometry and then deformed in consecutive frames to follow motion trajectories. Morphing can obviously result in artifacts, which the authors avoid by blurring morphed frames and then sharpening fully rendered frames. In that respect, the method takes advantage of similar perceptual limitations as TRM and NCSFI. Reprojection methods (Didyk et al.[11], ASW [2]), however, are much more complex than TRM and require a motion field, which could be expensive to compute, reducing the performance saving from 50%. Such methods have limitations handling transparent objects, specularities, disocclusions, changing illumination, motion discontinuities and complex motion parallax. We argue that rendering a frame at a reduced resolution (as done in TRM) is both a simpler and more robust alternative. Although minor loss of contrast could occur around high-contrast edges such as in Figure 6; in Section 8 we demonstrate that the failures of a state-of-the-art reprojection

Table 1: Comparison of alternative techniques. For detail, please see text in Section 6.

	Peak luminance	Motion Blur	Flicker	Artifacts	performance saving
Full frame rate	100%	none	none	none	0%
Reprojection (ASW, Didyk et al.[10])	100%	reduced	none	reprojection artifacts	varies; 50% max.
Half frame rate	100%	strong	none	judder	50%
Interlace	50%	reduced	moderate	combing	50%
BFI	50%	reduced	severe	none	50%
NCSFI	100%	reduced	mild	ghosting	50%
TRM (our)	100%	reduced	mild	minor	37–49%

technique, ASW, produce much less preferred results than TRM. Moreover, reprojection cannot be used for efficient transmission as it would require transmitting motion fields, thus eliminating potential bandwidth savings.

6.1 Fourier analysis

To further distinguish our approach from previous methods, we analyze each technique using the example of a vertical line moving with constant speed from left to right. We found that such a simplistic animation provides the best visualization and poses a good challenge for the compared techniques. Figure 12 shows how a single row of such a stimulus changes over time when presented using different techniques. The plot of position vs. time forms a straight line for a real-world motion, which is not limited by frame rate (top row, 1st column). But the same motion forms a series of vertical line segments on a 60 Hz OLED display, as the pixels must remain constant for 1⁄60-th of a second. When the display frequency is increased to 120 Hz, the segments become shorter. The second column shows the stabilized image on the retina assuming that the eye perfectly tracks the motion. The third column shows the image integrated over time according to the Talbot-Plateau law.

60 Hz animation appears more blurry than the 120 Hz animation (see 3rd column) mostly due to a hold-type blur. The three bottom rows compare three techniques aiming to improve motion quality, including ours. The black frame insertion (BFI) reduces the blur to that of 120 Hz without the need to render an image 120 frames per second, but it also reduces the brightness of an image by half. NCSFI [6] does not suffer from reduced brightness and also reduces hold-type blur, but to a lesser degree than BFI. Our technique (bottom row) has all the benefits of NCSFI but achieves stronger blur reduction, on par with the 120 Hz video.

Further advantages of our technique are revealed by analyzing the animation in the frequency domain. The fourth column in Figure 12 shows the Fourier transform of the motion-compensated image (2nd column). The blue diamond shape represents the range of visible spatial and temporal frequencies, following the stCSF shape from Figure 3-left. The perfectly stable *physical image* of a moving line (top row) corresponds to the presence of all spatial frequencies in the Fourier domain (the Fourier transform of a Dirac peak is a constant value). Motion appears blurry on a 60 Hz display and hence we see a short line along the x-axis, indicating the loss of higher spatial frequencies. More interestingly, there are a number of aliases of the signal in higher temporal frequencies. Such aliases reveal themselves as non-smooth motion (crawling edges). The animation shown on a 120 Hz display (3rd row) reveals less hold-type blur (longer line on the x-axis) and it also puts aliases further apart, making them potentially invisible. BFI and NCSFI result in a reduced amount of blur, but temporal aliasing is comparable to a 60 Hz display. Our method reduces the contrast of every second alias, thus making them much less visible. Therefore, although other methods can reduce hold-type blur, only our method can improve the smoothness of motion.

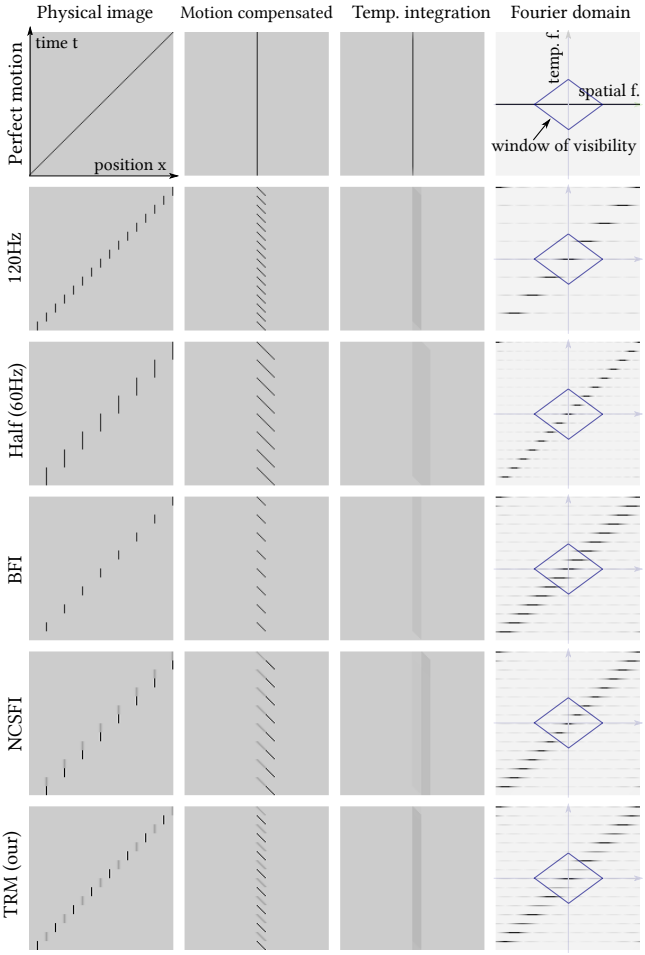

Figure 12: A simple animation consisting of a vertical line moving from left to right as seen in real-world (top row), and using different display techniques (remaining rows). The columns illustrate the physical image (1st column), the stabilized image on the retina (2nd column) and the image integrated by the visual system (3rd column). The 4th column shows the 2nd column in the Fourier domain, where the diamond shape indicates the range of spatial and temporal frequencies visible to the human eye.

7 Applications

In this section we demonstrate how TRM can benefit transmission, VR rendering and high-frame-rate monitors.

7.1 Transmission

One substantial benefit of our method is the reduced bandwidth of frame data that needs to be transmitted from a graphics card to the headset. Even current generation headsets, offering low angular

resolution, require custom high bandwidth links to send 90 frames per second without latency. Our method reduces that bandwidth by 37–49%. Introducing such coding would require an additional processing step to be performed on the headset (*Decoding & display* block in Figure 4). But, due to the simplicity of our method, such processing can be relatively easily implemented in hardware.

In order to investigate the potential for additional bandwidth savings, we tested our method in conjunction with one of the latest compression protocols designed for real-time applications — the JPEG XS standard (ISO/IEC 21122). The JPEG XS standard defines a low-complexity and low-latency compression algorithm for applications where (due to the latency requirements) it was common to use uncompressed image data [9]. As JPEG XS offers various degrees of parallelism, it can be efficiently implemented on a multitude of CPUs, GPUs and FPGAs.

We compared four JPEG compression methods: Lossless, XS bpp=7, XS bpp=5 and XS bpp=3, and computed the required data bandwidth for a number of TRM reduction factors. For this purpose we used four video sequences. As shown in Figure 13, the application of our method noticeably reduces bits per pixel (bpp) values for all four compression methods. Notably, frames compressed with JPEG XS bpp=7 and encoded with TRM with a reduction factor of 0.5 required only about 4.5 bpp, offering bandwidth reduction of more than one third, when compared with JPEG XS bpp=7. A similar trend can be observed for the remaining JPEG XS compression levels (bpp=5 and bpp=3). We carefully inspected the sequences that were encoded with both TRM and JPEG XS for the presence of any visible artifacts related to possible interference between coding and TRM, but were unable to find any distortions. This demonstrates that TRM can be combined with traditional coding to further improve coding efficiency for high-refresh-rate displays.

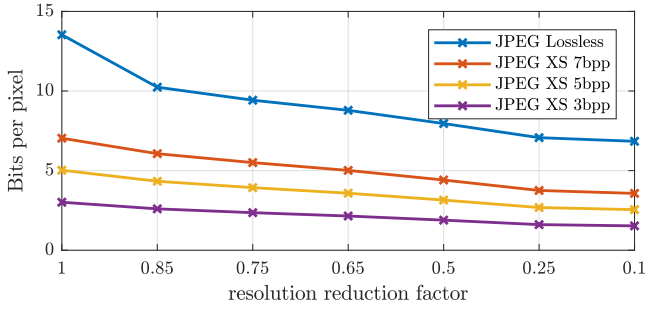

Figure 13: Required bandwidth of various image compression formats across selected TRM reduction factors.

7.2 Virtual reality

To better distribute rendering load over frames in stereo VR, we render one eye at full resolution and the other eye at reduced resolution; then, we swap the resolutions of the views in the following frame. Such alternating binocular presentation will not result in higher visibility of motion artifacts than the corresponding monocular presentation. The reason is that the sensitivity associated with disparity estimation is much lower than the sensitivity associated with luminance contrast perception, especially for high and spatial and temporal frequencies [16]. Another important consideration is whether the fusion of low- and high-resolution frames happens before or after binocular fusion. The latter scenario, evidenced as the Sherrington effect [25], is beneficial for us as it reduces the flicker visibility as long as high- and low-resolution frames are presented to different eyes. The studies on binocular flicker [25] suggest that while most of the flicker fusion is monocular, there is also a measurable binocular component. Indeed, we observed that flicker is less visible in a binocular presentation on a VR headset.

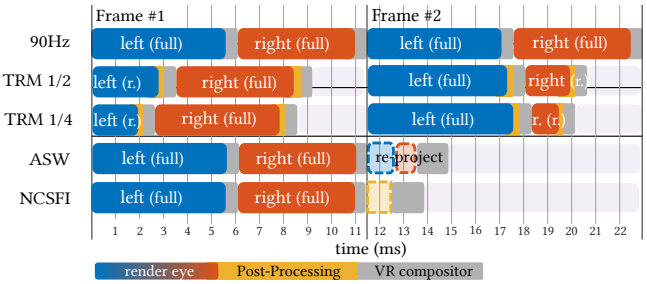

Figure 14: Measured performance of 90 Hz full resolution rendering on HTC Vive for two consecutive frames averaged over 1500 samples (top); compared with our TRM method with ½ and ¼ resolution reduction (center and bottom). Dashed lines (Frame 2 for ASW and NCSFI) indicate estimated time duration. Unutilized time periods can be used to load or compute additional visual effects or geometry.

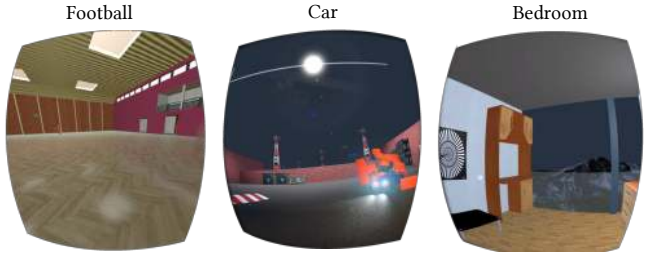

Figure 15: Stimuli used for validation in Experiments 2 and 3.

Reducing the resolution of one eye can reduce the number of pixels rendered by 37–49%, depending on the resolution reduction. We found that a reduction of ½ (37.5% pixel saving) produces good quality rendering on the HTC Vive headset. We measured the performance of our algorithm in a fill-rate-bound football scene (Figure 15 bottom) with procedural texturing, reflections, shadow mapping and per-fragment lighting. The light count was adjusted to fully utilize the 11ms frame time on our setup (HTC Vive, Intel i7-7700 processor and NVIDIA GeForce GTX 1080 Ti GPU). As Figure 14 indicates, we observed a 19-25% speed-up for an unoptimized OpenGL and OpenVR-based implementation. Optimized applications with ray tracing, hybrid rendering [27] and parallax occlusion mapping [32] could benefit even more.

A pure software implementation of TRM can be easily integrated into existing rendering pipelines as a post-processing step. The only significant change in the existing pipeline is the ability to alternate full- and reduced-resolution render targets. In our experience, available game engines either support resizeable render targets or allow light-weight alteration of the viewport through their scripting infrastructure. When available, resizeable render targets are preferred to avoid MSAA resolves in unused regions of the render target.

7.3 High-frame-rate monitors

The same principle can be applied to high-frame-rate monitors commonly used for gaming. The saving from resolution reduction could be used to render games at a higher quality. The technique could also be potentially used to reduce bandwidth for transmission of HFR video from cameras. However, we noticed that the difference between 120 Hz and 60 Hz is noticeable mostly for very high angular velocities, such as those experienced in VR and first-person games. The benefit of high frame rates is more difficult to observe for traditional video content.

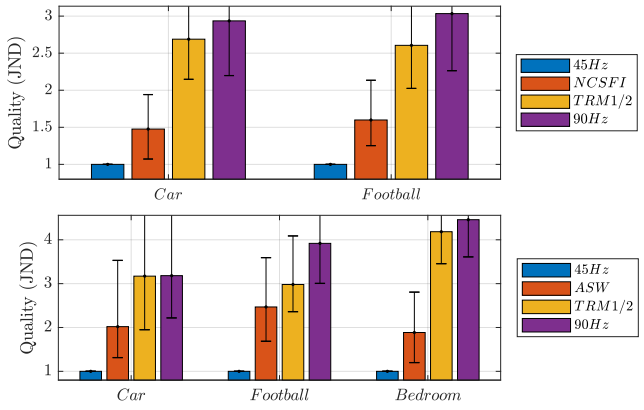

Figure 16: Results of experiment 2 on the HTC Vive (top) and experiment 3 on the Oculus Rift (bottom). Error bars denote 95% confidence intervals.

8 Experiments 2 and 3: Validation in VR

The final validation of our technique is performed in Experiments 2 and 3, comparing TRM with baseline rendering and two alternative techniques: NCSFI and state-of-the-art reprojection (ASW).

Setup:

We validated the technique on two different VR headsets running on 90 Hz – HTC Vive and Oculus Rift CV1 for Expriment 2 and 3 respectively. ASW is not implemented for the HTC Vive, so in Experiment 2 we only tested TRM against baseline renderings and NCSFI. In Experiment 3 we replaced NCSFI with the latest ASW implementation on the Oculus Rift. We used the same PC as in Experiment 1. The participants were asked to perform the experiment on a swivel chair and were encouraged to move their heads around.

Stimuli:

In each trial the observer was placed in two brief (10s each) computer-generated environments, identical in terms of content, but rendered using one of the following five techniques: (1) 90 Hz full refresh rate, (2) 45 Hz halved refresh rate, duplicating each frame (3) TRM with a ½ down-sampled render target for every other frame (4) nonlinearity compensated smooth frame insertion (NCSFI) in the HTC Vive Session, (5) Asynchronous Spacewarp (ASW) in the Oculus Rift session. Because NCSFI was not meant to be used in VR rendering, we had to make a few adaptations: to save on rendering time, only every other frame was rendered. These frames were used to create sharpened and blurry frames in accordance with the original design of the algorithm. For this comparison, we used the same blur method as for TRM, focusing only on the two fundamental differences between NCSFI and TRM: (1) NCSFI duplicates frames and (2) residuals are always added from sharp to blurry frames, regardless of motion. For ASW the content was rendered at 45 Hz and intermediate frames were generated using Oculus' implementation of ASW.

The computer-generated environments (Figure 15) consisted of an animated *football*, a *car* and *bedroom* (used only in Experiment 3). The first two scenes encouraged the observers to follow motion; the last one was designed to challenge screen-space warping. These scenes were rendered using the Unity game engine.

Task:

Participants were asked to select the rendered sequence that had *better visual quality* and *motion quality*. Participants were presented with two techniques sequentially (10s each), with unlimited time afterwards to make their decisions. Before each session, participants were briefed about their task both verbally and in writing. For those participants who had never used a VR headset before, a short session was provided, where they could explore Valve's SteamVR lobby in order to familiarize themselves with the fully immersive environment. We used a pairwise comparison method with a full design, in which all combinations of pairs were compared.

Participants:

Nine paid participants aged 18–40 with normal or corrected-to-normal vision took part in Experiments 2 and 3. The majority of participants had little or no experience with virtual reality.

Results:

The results of the pairwise comparison experiments were scaled using publicly available software[1] under Thurstone Model V assumptions in just-objectionable differences (JODs), which quantify the relative quality differences between the techniques. A difference of 1 JOD means that 75% of the population can spot a difference between two conditions. The details of the scaling procedure can be found in [24]. Since JOD values are relative, the 45 Hz condition was fixed at 1 JOD for better presentation.

The results from experiment 2 shown at top of Figure 16 indicate that the participants could not observe much difference between our method and the original 90 Hz rendering. The NCSFI method improved slightly over the repeated frames (45 Hz) but was much worse than TRM or full-resolution rendering (90 Hz). We suspect that this is because of the strong ghosting artifacts, which were well visible when blurred frames were displayed out of phase with misaligned residuals for fast head motion. Note that the technique by Didyk et al. [11], although not tested in our experiment, uses the same strategy as NCSFI for handling under- and over-shoots.

The results from experiment 3 on the Oculus Rift, shown at the bottom of Figure 16, resemble the results of experiment 2 on the HTC Vive: the participants could not observe much difference between our method and a full 90 Hz rendering. ASW was seen to perform best in the *football* scene, whereas it performed worse in the *car* and *bedroom* scenes. This is because complex motion and color variations in these scenes could not be compensated with screen-space warping, resulting in well visible artifacts.

9 Limitations

TRM is applicable only to high-refresh-rate displays, capable of showing 90 or more frames per second. At lower frame rates, flicker becomes visible. TRM is most beneficial when the angular velocities of movement are high, such as those introduced by camera motion in VR or first-person games. Our technique requires characterization of the display on which it is used, as explained in Section 4.4. This is a relatively easy step for OLED-based VR headsets, but the characterization is more involved for LCD panels. Unlike reprojection techniques, we need to render intermediate frames. This requires processing the full geometry of the scene every frame, which reduces performance gain for scenes that are not fragment-bound. However, this cost is effectively amortized in VR stereo rendering, as explained in Section 7.2. The method also adds to the memory footprint as it requires additional buffers, one for storing the previous frame and another one for the residual. The memory footprint, however, is comparable to or smaller than that of reprojection methods.

10 Conclusions

The visual quality of VR and AR systems will improve with increased display resolution and higher refresh rates. However, rendering such a large number of pixels with minimum latency is challenging even for high-end graphics hardware. To reduce the GPU

[1] pwcmp - `https://github.com/mantiuk/pwcmp`

workload and the data sent to the display, we propose the Temporal Resolution Multiplexing algorithm. TRM achieves a significant speed-up by requiring only every other frame to be rendered at full resolution. The method takes advantage of the limited ability of the visual system to perceive details of high spatial and temporal frequencies and renders a reduced number of pixels to produce smooth motion. TRM integrates easily into existing rasterization pipelines, but could also be a natural fit with any fill-rate-bound high-frame-rate application, such as real-time ray tracing.

10.1 Future work

TRM could be potentially beneficial at lower frame rates (<90 Hz) if flickering artifacts could be avoided. We would like to explore the possibility of predicting flickering and selectively attenuating temporal contrast that causes flicker. Since color vision is significantly limited at high spatio-temporal frequencies, TRM could be combined with chroma sub-sampling. However, it is unclear whether rendering mixed resolution luma and chroma channels can reduce rendering load.

ACKNOWLEDGEMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n° 725253–EyeCode) and EPSRC EP/N509620/1
We thank Marcell Szmandray for his panorama images.

REFERENCES

[1] W. Allen and R. Ulichney. Wobulation: Doubling the addressed resolution. 2012.

[2] D. Beeler, E. Hutchins, and P. Pedriana. Asynchronous spacewarp. `https://developer.oculus.com/blog/asynchronous-spacewarp/`, 2016. Accessed: 2018-05-02.

[3] R. S. Berns. Methods for characterizing CRT displays. *Displays*, 16(4 SPEC. ISS.):173–182, may 1996.

[4] F. Berthouzoz and R. Fattal. Resolution enhancement by vibrating displays. *ACM Transactions on Graphics*, 31(2):1–14, apr 2012.

[5] D. H. Brainard. The psychophysics toolbox. *Spatial vision*, 10:433–436, 1997.

[6] H. Chen, S.-s. Kim, S.-h. Lee, O.-j. Kwon, and J.-h. Sung. Nonlinearity compensated smooth frame insertion for motion-blur reduction in LCD. In *2005 IEEE 7th Workshop on Multimedia Signal Processing*, pages 1–4. IEEE, oct 2005.

[7] S. Daly, N. Xu, J. Crenshaw, and V. J. Zunjarrao. A Psychophysical Study Exploring Judder Using Fundamental Signals and Complex Imagery. *SMPTE Motion Imaging Journal*, 124(7):62–70, oct 2015.

[8] S. J. Daly. Engineering observations from spatiovelocity and spatiotemporal visual models. In B. E. Rogowitz and T. N. Pappas, editors, *Human Vision and Electronic Imaging*, volume 3299, pages 180–191, jul 1998.

[9] A. Descampe, J. Keinert, T. Richter, S. Fossel, and G. Rouvroy. Jpeg xs, a new standard for visually lossless low-latency lightweight image compression. In *Applications of Digital Image Processing XL*, pages 10396 – 10396, 2017.

[10] P. Didyk, E. Eisemann, T. Ritschel, K. Myszkowski, and H.-P. Seidel. Apparent display resolution enhancement for moving images. *ACM Transactions on Graphics*, 29(4):1, jul 2010.

[11] P. Didyk, E. Eisemann, T. Ritschel, K. Myszkowski, and H. P. Seidel. Perceptually-motivated real-time temporal upsampling of 3D content for high-refresh-rate displays. *Computer Graphics Forum*, 29(2):713–722, 2010.

[12] X.-f. Feng. LCD motion-blur analysis, perception, and reduction using synchronized backlight flashing. In *Human Vision and Electronic Imaging*, volume 6057, pages 1–14, 2006.

[13] A. Fujibayashi and Choong Seng Boon. Application of motion sharpening effect in video coding. In *2008 15th IEEE International Conference on Image Processing*, pages 2848–2851. IEEE, 2008.

[14] E. Hartmann, B. Lachenmayr, and H. Brettel. The peripheral critical flicker frequency. *Vision Research*, 19(9):1019–1023, 1979.

[15] M. Hirsch, G. Wetzstein, and R. Raskar. A compressive light field projection system. *ACM Trans. Graph.*, 33(4):58:1–58:12, July 2014.

[16] D. M. Hoffman, V. I. Karasev, and M. S. Banks. Temporal presentation protocols in stereoscopic displays: Flicker visibility, perceived motion, and perceived depth. *Journal of the Society for Information Display*, 19(3):271, 2011.

[17] I. Kauvar, S. J. Yang, L. Shi, I. McDowall, and G. Wetzstein. Adaptive color display via perceptually-driven factored spectral projection. *ACM Transactions on Graphics*, 34(6):1–10, oct 2015.

[18] D. H. Kelly. Motion and vision II Stabilized spatio-temporal threshold surface. *Journal of the Optical Society of America*, 69(10):1340, oct 1979.

[19] D. H. Kelly. Retinal inhomogeneity I Spatiotemporal contrast sensitivity. *Journal of the Optical Society of America A*, 1(1):107, jan 1984.

[20] J. Laird, M. Rosen, J. Pelz, E. Montag, and S. Daly. Spatio-velocity CSF as a function of retinal velocity using unstabilized stimuli. In B. E. Rogowitz, T. N. Pappas, and S. J. Daly, editors, *Human Vision and Electronic Imaging*, volume 6057, page 605705, feb 2006.

[21] P. Lincoln, A. Blate, M. Singh, T. Whitted, A. State, A. Lastra, and H. Fuchs. From Motion to Photons in 80 Microseconds: Towards Minimal Latency for Virtual and Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1367–1376, apr 2016.

[22] J. Mulligan. Methods for spatiotemporal dithering. *SID International Symposium Digest of Technical ...*, pages 1–4, 1993.

[23] D. Nehab, P. V. Sander, J. Lawrence, N. Tatarchuk, and J. R. Isidoro. Accelerating Real-time Shading with Reverse Reprojection Caching. *Proc of Symposium on Graphics Hardware*, pages 25–35, 2007.

[24] M. Perez-Ortiz and R. K. Mantiuk. A practical guide and software for analysing pairwise comparison experiments. *arXiv preprint*, dec 2017.

[25] F. H. Perrin. A Study in Binocular Flicker. *Journal of the Optical Society of America*, 44(1):60, jan 1954.

[26] M. Pettineo. Rendering The Alternate History of The Order: 1886. Presented at Advances in Real-Time Rendering in Games course at SIGGRAPH 2015, 2015.

[27] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan. Ray tracing on programmable graphics hardware. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[28] D. A. Robinson, J. L. Gordon, and S. E. Gordon. A model of the smooth pursuit eye movement system. *Biological cybernetics*, 55(1):43–57, 1986.

[29] B. Sajadi, M. Gopi, and A. Majumder. Edge-guided resolution enhancement in projectors via optical pixel sharing. *ACM Transactions on Graphics*, 31(4):1–122, jul 2012.

[30] D. Scherzer, L. Yang, O. Mattausch, D. Nehab, P. V. Sander, M. Wimmer, and E. Eisemann. Temporal coherence methods in real-time rendering. *Computer Graphics Forum*, 31(8):2378–2408, 2012.

[31] T. Takeuchi. Sharpening image motion based on the spatio-temporal characteristics of human vision. *Human Vision and Electronic Imaging*, 5666(March 2005):83–94, 2005.

[32] N. Tatarchuk. Practical parallax occlusion mapping with approximate soft shadows for detailed surface rendering. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 81–112, New York, NY, USA, 2006. ACM.

[33] A. Vlachos. Advanced VR Rendering Performance. In *Game Developers Conference (GDC)*, 2016.

[34] A. B. Watson and A. J. Ahumada. The pyramid of visibility. In *Human Vision and Electronic Imaging*, volume 2016, pages 1–6, feb 2016.

[35] A. B. Watson, A. J. Ahumada, and J. E. Farrell. Window of visibility: a psychophysical theory of fidelity in time-sampled visual motion displays. *Journal of the Optical Society of America A*, 3(3):300, mar 1986.

[36] J. H. Westerink and C. Teunissen. Perceived sharpness in moving images. In B. E. Rogowitz and J. P. Allebach, editors, *Human Vision and Electronic Imaging*, pages 78–87, oct 1990.