Joint Virtual Reality Conference of ICAT - EGVE - EuroVR (2012)
R. Boulic, C. Cruz Neira, K. Kiyokawa, and D. Roberts (Editors)

Fast Motion Rendering for Single-Chip Stereo DLP Projectors

M. Lancelle[1], G. Voß[1] and D. W. Fellner[1,2]

[1]Fraunhofer IDG@NTU, Singapore
[2]GRIS, TU Darmstadt & Fraunhofer IGD, Germany

Figure 1: An object moves fast across the screen. The top row shows the type of image that is sent to the projector. The bottom row shows photos with 1/50s exposure time, with the camera following the object motion, similar to a human eye tracking the object. a) and d) show conventional rendering at 60 frames/second (fps). A double image with color fringes is the result. b) and e) show how motion blur reduces these artefacts, however, the object is blurred. c) and f) show our approach with 120 fps and temporal correction per color, leading to a much sharper object without color fringes.

Abstract

Single-chip color DLP projectors show the red, green and blue components one after another. When the gaze moves relative to the displayed pixels, color fringes are perceived. In order to reduce these artefacts, many devices show the same input image twice at the double rate, i.e. a 60 Hz *source image is displayed with* 120 Hz.
Consumer stereo projectors usually work with time interlaced stereo, allowing to address each of these two images individually. We use this so called 3D mode for mono image display of fast moving objects. Additionally, we generate a separate image for each individual color, taking the display time offset of each color component into account. With these 360 *images per second we can strongly reduce ghosting, color fringes and jitter artefacts on fast moving objects tracked by the eye, resulting in sharp objects with smooth motion.*
Real-time image generation at such a high frame rate can only be achieved for simple scenes or may only be possible by severely reducing quality. We show how to modify a motion blur post processing shader to render only 60 frames/second *and efficiently generate good approximations of the missing frames.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Introduction

In many settings, projectors are a desirable type of display. However, they still suffer from problems such as low luminance and high lamp replacement cost, preventing a wider use. However, there is a growing market for home cinema and gaming. A recent interest in 3D movies brought time interlaced stereo capability to some entry level consumer projectors with NVIDIA's 3D Vision [NVI]. The users wear LC shutter glasses to see the correct image with each eye. These projectors also allow us to improve the display of fast moving objects.

DOI: 10.2312/EGVE/JVRC12/029-036

Such objects moving on a display screen with a relative speed of more than 1 pixel/frame may appear blurred and lead to slightly visible stuttering motion. This is especially the case for objects with predictable motion that are tracked by the human eye. On a typical consumer monitor or projector this effect is well visible, even with 60 frames/second.

Frame interpolation has gained attention with growing image processing power available on consumer displays and high definition TVs. Also here, the display of fast motion, e.g. in sports footage, suffers from the relatively low update rates. This effect is sometimes reduced by intelligent processing in the display device that insert one or more interpolated frames per source frame. However, artefacts occur especially at object borders. In addition, it seems to be much a personal preference whether the presence of interpolated frames is regarded as pleasing or not.

In Virtual Reality image generation, additional information can be stored along with the frame buffer and can be used to reduce artefacts. For offline production, videos can be rendered with 360 frames/second and easily converted into a motion color corrected 120 Hz mono or 60 Hz stereo movie. The playback of this preprocessed movie on a matching projector results in much sharper moving objects and smoother motion. For real time rendering, we also explore how to reduce computation cost by using a shader similar to a motion blur shader in order to generate six frames out of one source frame.

2. Related Work

2.1. DLP Projectors and Modifications

Digital Light Processing (DLP) projectors use deformable mirrors to influence the direction of the reflected light for each pixel [Hor90]. Variations in brightness are achieved via pulse width modulation with high speed binary patterns. Single-chip projectors show the colors sequentially, one after another. With an incandescent light source, a color wheel (see Figure 3b) is used to only pass through the respective color. LED (or Laser) projectors can simply turn on the required light source at the correct time and do not need a color wheel. Usually, single-chip DLPs are used in entry level consumer products. In contrast, professional DLP projectors with a high light throughput often use three chips, one for each color component, displaying each color at the same time. Our technique requires single chip stereo projectors, as it takes advantage of the time offsets between the colors. Furthermore, the images for each color wheel cycle must addressed individually, which means that it must be a 3D (or 120 Hz capable) device.

Raskar et al. removed the color wheel and modified the firmware of a single-chip DLP projector to project patterns imperceptible to the human eye but visible to a synchronized camera image for 3D acquisition [RWC*98]. Cotting et al. achieve this without projector modifications by measuring and cleverly using mirror flip sequence timing [CNGF04]. Nguyen et al. also removed the color wheel of such a projector for high speed real time 3D capturing [NW10]. Multi viewer stereo display setups likewise profit from the time interlaced signals and use multiple projectors, modified color filters and glasses [MBCS01], [KKB*11].

2.2. Temporal Frame Interpolation

Temporal frame interpolation is used for high quality frame rate conversion, e.g. for slow motion special effects or to avoid the telecine (2–3 pull-down) problems of judder or mixed frames. Examples of such algorithms are presented in [AtLBCA02] and [WPUB11].

Frame interpolation is also commonly used in video compression [LG91], [KKJK99]. The idea is to transmit frames at a lower frame rate. The missing intermediate frames are generated by interpolation. For those frames, only the difference to the interpolated image is transmitted, usually resulting in higher compression. In some cases, this is block based and does not necessarily represent actual motion but may point to a different but similar looking region in the image.

High resolution TV displays can often display at higher frame rates than what the source material offers. In some models, one or three intermediate frames are generated for each source frame for a smoother motion (see Figure 2d, here shown for a single-chip DLP projector). The filtering also includes a motion blur reduction. However, artefacts at object boundaries are common and let many users disable this feature.

2.3. Frame Duration

Another possibility for smoother motion is *black frame insertion* (see Figure 2g), equivalent to the 180 degree shutter that is traditionally used in movie projectors. The light source can be modulated to achieve this effect, fast enough to avoid flickering. A combination of frame interpolation and black frame insertion is possible (see Figure 2h).

2.4. High Frame Rate Movies

Video capture systems for very high frame rates exist but are mostly used for inspection or special effects slow motion footage. At the time of writing (2012), the movie and projector industries prepare for high frame rates (HFR), meaning 48 and 60 full frames/second (fps) (Christie, [CDSU12]) or 96 fps and 2×60 fps (Sony). The intentions are to avoid problems with camera motion for 3D movies, to enable the use of faster camera motion and to provide a quality advantage compared to watching movies at home. A low cost replacement of the integrated media board in the projectors is promising for a wide adoption of the technology in the cinemas.

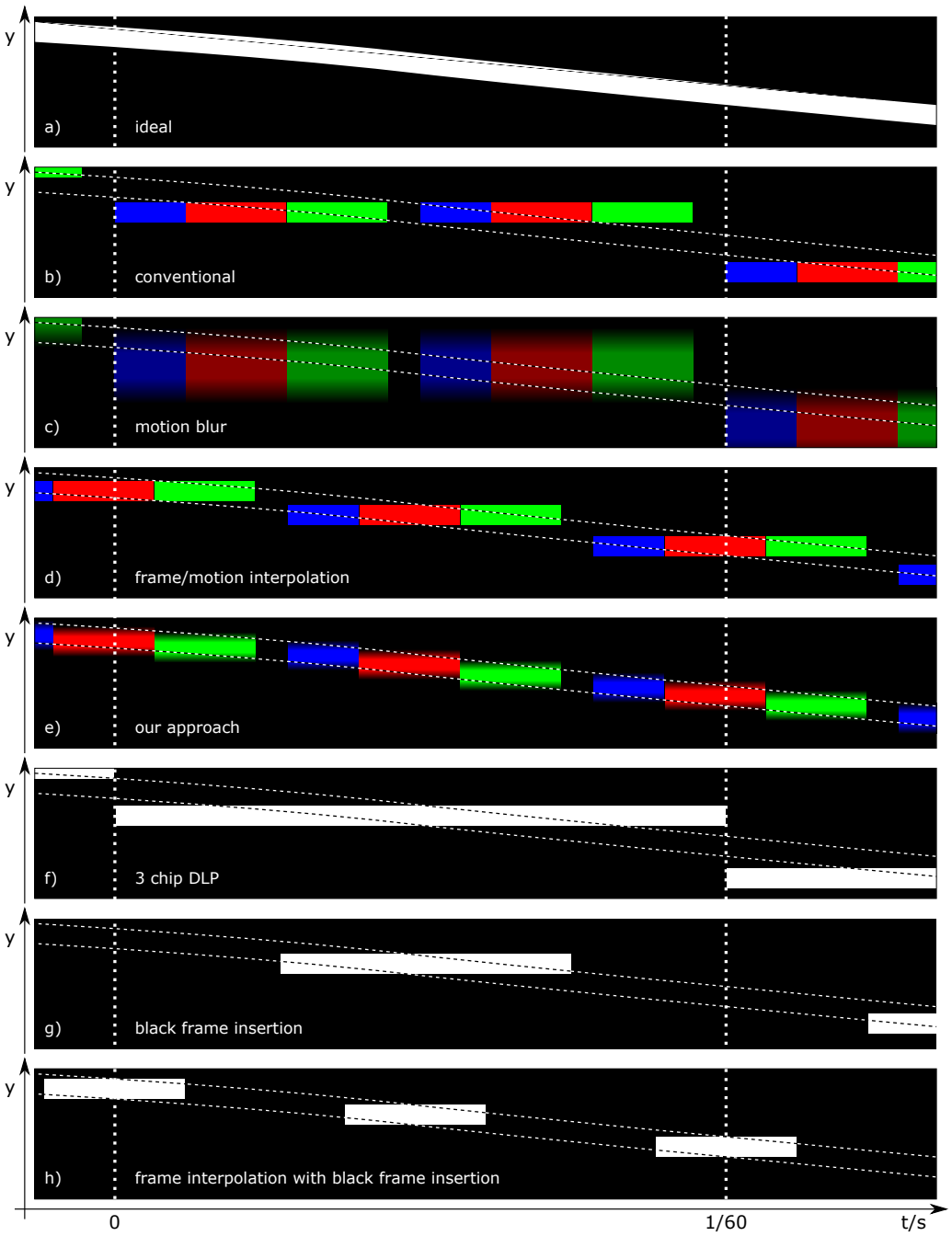

Figure 2: A white object moving over time. The ideal motion is shown in a) and is indicated by the dotted lines in b) to h). The deviation of this ideal motion shows the amount of ghosting and color fringes. b) Conventional rendering with 60 fps on a typical single-chip DLP projector. c) With motion blur. d) $1\times$ frame interpolation. e) Our approach. f) Three-chip DLP showing colors simultaneously. g) Black frame insertion. h) A combination of $1\times$ frame interpolation and black frame insertion.

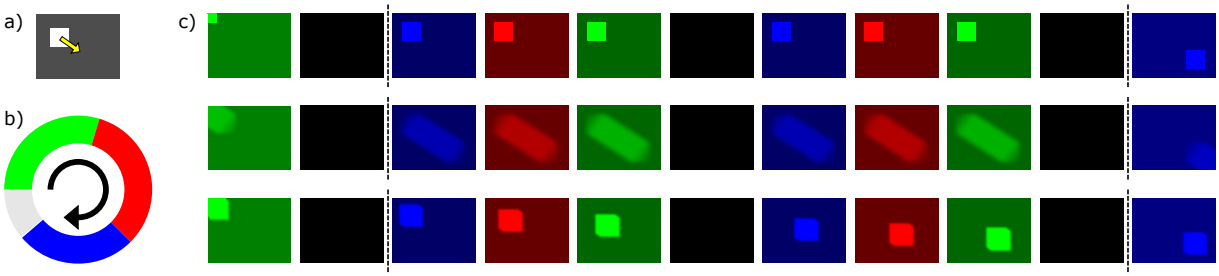

Figure 3: a) A white square moving fast to the bottom right, to be displayed on a projector. b) A typical color wheel of a single-chip DLP projector. c) Display options on such a projector. Each color channel is displayed one after another (from left to right). The black frames represent the here unused clear segment. Top row: conventional rendering without motion blur. With a 60 Hz source rate displayed at 120 Hz the object jumps. Middle row: With motion blur, the object appears very blurred. Bottom row: our approach on a 60 Hz stereo projector providing 120 Hz in mono, also taking the color timing into account, resulting in a much sharper object with smooth motion.

2.5. Motion Blur

Computer generated images are usually rendered for a single moment in time. For animation sequences or sometimes for still images, motion blurred images are computed. This can be an artistic choice, mimicking a real camera, or can be used to avoid stuttering motion artefacts. Motion blur can easily be integrated in a stochastical renderer, spreading samples over the exposure time. Another approach is to compute many complete frames during the exposure time and use the average of all frames as result. However, in both cases rendering time may increase significantly when using enough samples to avoid ghosting artefacts or too large amounts of noise. In order to reduce the cost for motion blur, faster 2D approximations are often used, such as in [Neu07] by Neulander et al. In addition to the color buffer, their algorithm uses a per-pixel velocity buffer and the depth buffer which are generated during the 3D rendering. The color buffer values are spread along a linear path according to the velocity buffer, as long as they are not covered by pixels which are in front. Ghosting effects can be partially compensated by jittering the animation time per pixel. A similar technique can also be used for real-time rendering. A recent example is the algorithm by McGuire et al. [MHBO12] who manage to quickly produce images with little artefacts. They convert the scatter algorithm into an efficient gather algorithm by first finding the maximum velocity in the neighborhood of each pixel. Along this direction, all samples are collected that are projected onto that pixel and are not behind the actual surface (see Section for details). Even though there are some approximations and restricting assumptions, the results are impressive.

Likewise, stop motion movies profit from artificial motion blur that is added in post production [BE01]. As velocity and depth information are not available, they must be estimated using optical flow and other heuristics.

2.6. Constant Frame Rate Rendering

Our algorithm only works when the scene can be rendered at a sufficient frame rate. In our case we ultimately need 360 fps. Binks et al. show how to achieve an optimized quality rendering with a constant frame rate by rendering to an intermediate buffer of a variable size, called Dynamic Resolution Rendering [Bin11]. When the rendering of the content is fast, a larger viewport is used and downsampled to the frame buffer for antialiasing. When rendering takes long, the viewport size is decreased to compensate for the slow rendering and the image is upscaled to fill the frame buffer.

2.7. Spatial Subsampling

ClearType by Microsoft [Pla00], [PKH*00] is used for font rendering on LCD screens. It takes the subpixel structure of the monitor into account to reduce stair case artefacts on edges. The basic idea is very much related to our approach, but while ClearType operates in the spatial domain, we employ temporal subsampling.

3. Rendering at 360 fps vs. 60 fps

When the gaze moves relative to the image of a single-chip DLP projector, color fringes are perceived at edges in the content or at the screen edges, so called rainbow artefacts. To reduce such artefacts, the complete color cycle is usually displayed at least twice during the frame time. However, when the eye tracks a moving object, this leads to ghosting in addition to the rainbow effect, as one image is always behind and one in front of the actual position. Three-chip movie projectors also show each frame twice or three times to avoid flickering and also suffer from ghosting, thereby limiting the type of motion in the content which can be displayed without this artefact.

While the movie industry may move from 24 fps to 48 or 60 fps, does it really make sense to use a significantly higher frame rate? We argue that indeed, especially with higher resolution it makes sense because the same motion results in a higher speed per pixel at a higher resolution. As an example, a horizontal camera pan with 6 pixels per $1/60$ s on the screen at a resolution of 1280×720 means an image feature is visible for 3.5s, clearly long enough to be tracked by the human eye. Motion blur can hide these artefacts but the objects appear blurred. At 360 fps the speed is 1 pixel per $1/60$ s, the scene appears considerably sharper and color fringes disappear on moving objects tracked by the eye (see Figure 1).

For time sequential stereo projection, entry level consumer projectors can be used to display 2×60 images/second or directly 120 images/second. Many color wheels include a small clear segment for increased brightness in presentations (see Figure 3b). However, the intensity for this channel is generated internally out of the RGB information. This option, also known as white boost, can often be turned off for better color reproduction. In this case, each individual displayed color channel can be addressed and used to display 360 images/second.

3.1. Motion Blur

Fast moving objects should exhibit motion blur for the duration of their display, i.e. $1/360$ s in our case. However, very fast objects, especially repeating patterns like fence structures or a rotating wheel that are not tracked by the eye any more, should not be color corrected. As the eye does not move along the object motion, this would introduce color fringes rather than avoiding them. As this depends on the scene content, we suggest to empirically test and use a maximum velocity threshold to apply a color offset. If the velocity for the individual pixel is above the threshold, standard motion blur with $1/120$ s can be used.

3.2. Exact Timing

So far we used the simplifying assumption that each color is shown for the same duration of $1/360$ s. However, as already mentioned, color wheels usually contain clear segments and show colors for different durations (again, see Figure 3b). This can be taken into account by adjusting the time offset accordingly. Also, as shown in [CNGF04], mirror flip sequences may be biased leading to a temporal shift especially for medium brightness values. This is harder to compensate for, as the output value determines the time offset to compute this value. In our projector this does not seem to be the case. In some of our tests we only use the simplified assumption of evenly spaced offsets and achieve reasonably good results.

To find the timing information for our projector, we did not want to open the case and measure the color wheel segments. Instead, we take a photo of the projector light, quickly rotating the camera (see Figure 4). The offsets can easily be measured with sufficient accuracy.

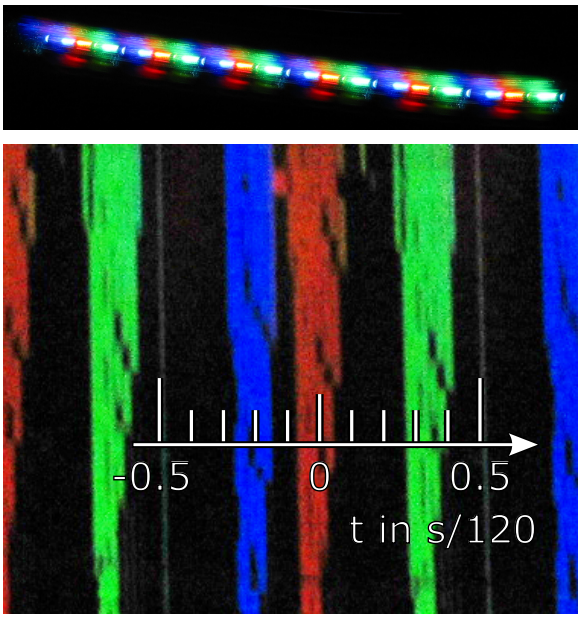

Figure 4: Photos of the projector light to find the color timing, quickly moving/rotating the camera from the right to the left, here with $1/20$ s exposure time. Top: A photo of the projector lens. Bottom: Photo of a thin vertical screen illuminated by the projector with a gray gradient reveals the mirror flip sequence patterns. The color offsets can easily be read out to sufficient accuracy, here 0.0, 0.32 and -0.2 times the frame duration for red, green and blue respectively.

Our projector, an Optoma GT750, shows first blue, then red, then green. It employs the NVision stereo system by NVidia and shows the image for the left eye first.

4. Implementation

Depending on the system setup, it should be either possible to directly display images with 120 fps or alternatively to show stereo images with 60 fps.

4.1. Offline Rendering

For computer generated, precomputed imagery like animated movies, the frame rate can easily be increased to 360 Hz. As mentioned, a simple approximation is to render with evenly spaced 360 frames/second which in practice shows to be sufficiently good. In case the rendering times are too long, a 2D motion blur algorithm can be adapted as discussed below. From the resulting images, only the respective color component is used and in case of stereo rendering written to the left and right image respectively. We wrote

simple AviSynth (http://avisynth.org/) scripts to convert a movie with a 360 fps input to either a 120 fps movie or a 60 fps stereo movie with our technique applied. With our consumer projector, we can watch a movie with 1280×720 pixels at 360 fps.

Compared to a 60 fps movie, twice the number of images must be stored. The amount of motion is smaller, which should lead to a slightly better compression. The color fringes on the edges of moving objects may have the adverse effect.

4.2. Real Time Rendering

We have tested several implementations, each with different advantages and drawbacks.

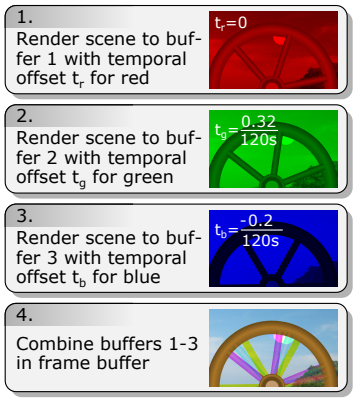

Figure 5: Simple rendering algorithm for 120 Hz output without motion blur, using four passes. First, each color channel is rendered using the respective time offset for the animation. Finally, the channels are combined and written to the frame buffer.

4.2.1. Rendering With 360 fps

A simple algorithm is shown in Figure 5. For each frame to be displayed with 120 fps, we render three images, each with the respective color and timing offset. The final forth pass combines the three channels and writes them to the frame buffer. While we use an OpenGL frame buffer object with three color attachments, the algorithm can also be implemented using the accumulation buffer. Note that the depth buffer must also be cleared for each of the first three passes. This method is the simplest to implement. However, rendering time increases by over 500 %, so complex scenes may not be rendered fast enough. In addition, motion blur is missing. However, it could be added e.g. by rendering even more frames and using the accumulation buffer to compute the average frames.

Variation. In scenes with a large amount of static or slowly moving objects, it may save time to render these objects at 60 Hz, writing the same color to six color buffers simultaneously and afterwards render the fast moving objects separately into each buffer. The result are approximations which hardly deviate from the true image. For fast moving objects with low depth complexity it may suffice to disable writing to the depth buffer in these six passes. Otherwise, the depth buffer must be saved and restored between each pass. The eighth and last pass again combines the channels and copies the data to the left and right frame buffer.

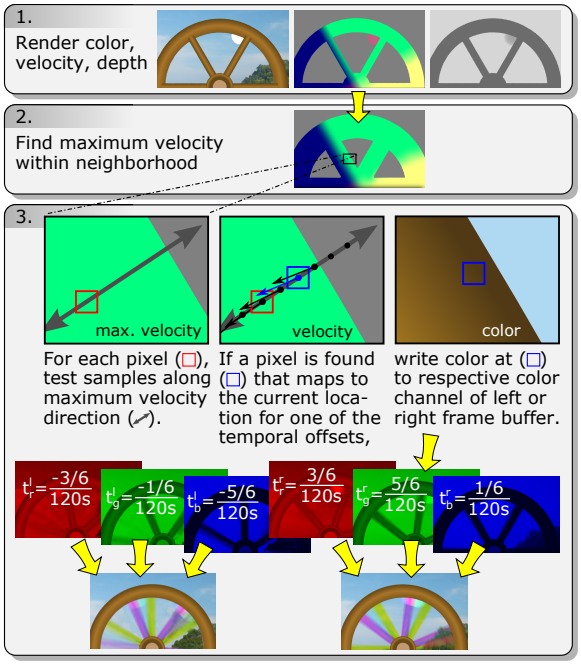

Figure 6: A modified motion blur algorithm. In the first pass, color and also velocity and linear depth buffers are generated. The second step computes the maximum velocity within a neighborhood, ignoring other velocities. In the final pass, for several samples at different animation times, pixels are searched along the maximum velocity direction that project to the current location. For conventional motion blur, the average of all samples is used as output. Our modification is to write the average of fewer samples in shorter time intervals to the respective output buffer (left/right) and color channel (see text for details). The output is a 60 Hz stereo image.

4.2.2. Modification of Motion Blur Algorithm

A much more involved approach is to use and modify a motion blur algorithm. For the pixel shader pipeline, the modification of output pixel locations is not possible in an efficient way. Thus, we follow the idea of the motion blur algorithm by McGuire et al. [MHBO12] and transform the scatter algorithm into a gather algorithm: Instead of computing the

new location for each source pixel, for each output pixel we search which source pixel maps to its location.

Rendering of the scene can again be done with 60 fps, this time also generating velocity and linear depth buffers in the first pass with OpenGL's multiple render targets (see Figure 6). In the second pass the maximum velocity in a neighborhood with a specified radius is computed. This radius depends on the maximum object velocity in screen space. In the final pass we gather color information along the direction of the previously found maximum velocity. This approximation drastically reduces the search space and works well with less complex velocity maps. The depth map is used to handle occlusions. Please refer to [MHBO12] for details in that particular algorithm.

For each output pixel, this computation is repeated for several animation times. For motion blur, the average color of all these samples is computed. For our algorithm, the only difference is that if the absolute velocity is below the user defined threshold, the values are not averaged but instead, six groups are averaged and written to the according buffers. Each of these groups corresponds to the display time of each color wheel segments of the left and right image. dAs an example, each group can average two samples, resulting in twelve samples taken into account.

Our modification adds only a negligibly small cost to the execution time of such a motion blur algorithm. According to McGuire et al. [MHBO12], their algorithm with 15 samples per pixel at 1280×720 takes 3.0 ms on a GeForce 480. This means using two samples per group should also render in 3 ms, leaving 13 ms for rendering the scene itself. Compared to the simple algorithm rendering the scene six times, this can reduce the total rendering time down to 20 % and may provide higher quality images due to motion blur.

We implemented a much simplified version of this algorithm as a proof of concept. It suffers from visible artefacts at object boundaries but demonstrates the concept.

5. Results and Future Work

Scrolling text, such as shown in Figure 1 is a good demonstration of our technique. We also made a mockup of a pinball game for one of our test scenes reffig:pinball, where the ball appears to move smoother with our technique. At very high speeds, the ball is not tracked by the eye and should get conventional motion blur.

On devices that do not match the requirements, our temporal color correction will increase the color fringes. This is also the case for screenshots and photos, where it should be turned off. Note that the same restrictions also apply e.g. for stereo displays or ClearType.

A key point in our idea to remove color fringes is that we need to know in which direction the eye moves and guess.

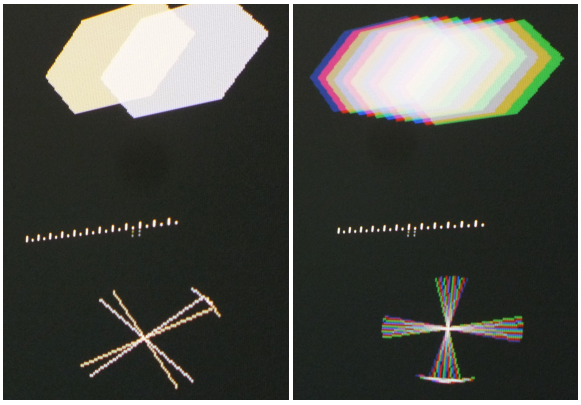

Figure 7: Photos of an early test. The exposure was set to $1/30$ s, capturing two full frames at 60 frames/second. Left: conventional rendering. Right: six times higher temporal resolution.

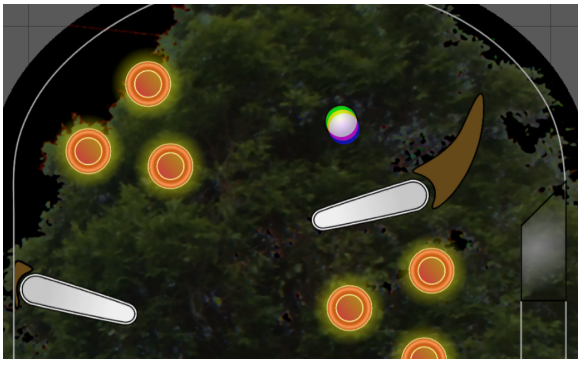

Figure 8: One of our test scenes is a mockup of a pinball game. We expect the ball to be tracked by the eye and apply temporal color correction.

Our assumption is that the case where the eye tracks an object is most important. If the eye moves differently, the color fringes on moving objects may appear more pronounced. As we leave the static objects unmodified, this is no problem in practice. An informal user study shows that in the few test cases our method is clearly preferred over the conventional display.

For stereoscopic rendering, the same technique can be applied, taking the offset between left and right image into account. However, we have not yet studied if users prefer this method and how this relates to symptoms of dizziness experienced by some people.

6. Conclusion

For each red, green and blue color wheel section we render one individual image, with the animation time set according to the average display time and thus taking the temporal

offset into account. We present an analysis and possible implementations for offline production and real time rendering. We show how to modify a 2D motion blur shader to save rendering time.

The benefit is a smooth motion with much sharper fast moving objects without color fringes. This also enables some applications that require the display of fast motion, such as games. We achieve this without any modification or additional hardware with an entry level consumer projector.

7. Acknowledgments

This research was partially done for Fraunhofer IDM@NTU, which is funded by the National Research Foundation (NRF) and managed through the multi-agency Interactive & Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore (MDA). We thank Volker Settgast for helping with initial tests and the reviewers for helpful comments.

References

[AtLBCA02] A T. L., TUNG LO B K., C J. F., A X. Z.: Frame interpolation scheme using inertia motion prediction, 2002. 2

[BE01] BROSTOW G. J., ESSA I.: Image-based motion blur for stop motion animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 561–566. URL: http://doi.acm.org/10.1145/383259.383325, doi:10.1145/383259.383325. 4

[Bin11] BINKS D.: Dynamic resolution rendering, 2011. URL: http://software.intel.com/en-us/articles/dynamic-resolution-rendering-article/. 4

[CDSU12] CHRISTIE DIGITAL SYSTEMS USA I.: High frame rates - how it works, 2012. URL: http://info.christiedigital.com/lp/how-hfr-works. 2

[CNGF04] COTTING D., NAEF M., GROSS M., FUCHS H.: Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality* (Washington, DC, USA, 2004), ISMAR '04, IEEE Computer Society, pp. 100–109. URL: http://dx.doi.org/10.1109/ISMAR.2004.30, doi:10.1109/ISMAR.2004.30. 2, 5

[Hor90] HORNBECK L. J.: Deformable-mirror spatial light modulators. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (May 1990), Efron U., (Ed.), vol. 1150 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 86–102. 2

[KKB*11] KULIK A., KUNERT A., BECK S., REICHEL R., BLACH R., ZINK A., FROEHLICH B.: C1x6: a stereoscopic six-user display for co-located collaboration in shared virtual environments. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, ACM, pp. 188:1–188:12. URL: http://doi.acm.org/10.1145/2024156.2024222, doi:10.1145/2024156.2024222. 2

[KKJK99] KUO T.-Y., KIM J., JAY KUO C.-C.: Motion-compensated frame interpolation scheme for h.263 codec. In *ISCAS* (1999). 2

[LG91] LE GALL D.: Mpeg: a video compression standard for multimedia applications. *Commun. ACM 34*, 4 (Apr. 1991), 46–58. URL: http://doi.acm.org/10.1145/103085.103090, doi:10.1145/103085.103090. 2

[MBCS01] MCDOWALL I. E., BOLAS M. T., CORR D., SCHMIDT T. C.: Single and multiple viewer stereo with dlp projectors. Woods A. J., Bolas M. T., Merritt J. O., Benton S. A., (Eds.), vol. 4297, SPIE, pp. 418–425. URL: http://link.aip.org/link/?PSI/4297/418/1, doi:10.1117/12.430854. 2

[MHBO12] MCGUIRE M., HENNESSY P., BUKOWSKI M., OSMAN B.: A reconstruction filter for plausible motion blur. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2012), I3D '12, ACM, pp. 135–142. URL: http://doi.acm.org/10.1145/2159616.2159639, doi:10.1145/2159616.2159639. 4, 6, 7

[Neu07] NEULANDER I.: Pixmotor: a pixel motion integrator. In *ACM SIGGRAPH 2007 sketches* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. URL: http://doi.acm.org/10.1145/1278780.1278808, doi:10.1145/1278780.1278808. 4

[NVI] Nvidia 3d vision. URL: http://www.nvidia.com/object/3d-vision-main.html. 1

[NW10] NGUYEN D. A., WANG Z.: High speed 3d shape and motion capturing system. In *ACM SIGGRAPH 2010 Posters* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 9:1–9:1. URL: http://doi.acm.org/10.1145/1836845.1836855, doi:10.1145/1836845.1836855. 2

[PKH*00] PLATT J. C., KEELY B., HILL B., DRESEVIC B., BETRISEY C., MITCHELL D. P., HITCHCOCK G., BLINN J. F., WHITTED T.: Displaced filtering for patterned displays. In *Proc. Society for Information Display Symposium* (2000), pp. 296–299. 4

[Pla00] PLATT J.: Optimal filtering for patterned displays. *Signal Processing Letters, IEEE 7*, 7 (july 2000), 179 –181. doi:10.1109/97.847362. 4

[RWC*98] RASKAR R., WELCH G., CUTTS M., LAKE A., STESIN L., FUCHS H.: The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 179–188. URL: http://doi.acm.org/10.1145/280814.280861, doi:10.1145/280814.280861. 2

[WPUB11] WERLBERGER M., POCK T., UNGER M., BISCHOF H.: Optical flow guided tv-l1 video interpolation and restoration. In *Proceedings of the 8th international conference on Energy minimization methods in computer vision and pattern recognition* (Berlin, Heidelberg, 2011), EMMCVPR'11, Springer-Verlag, pp. 273–286. URL: http://dl.acm.org/citation.cfm?id=2032617.2032641. 2