# Using a Camera to Capture and Correct Spatial Photometric Variation in Multi-Projector Displays

Aditi Majumder
Department of Computer Science
University of California, Irvine
majumder@ics.uci.edu

David Jones, Matthew McCrory, Michael E Papka, Rick Stevens
Mathematics and Computer Science Division
Argonne National Laboratory
{jones,mccrory,papka,stevens}@mcs.anl.gov

## Abstract

Photometric variation in multi-projector displays is arguably the most vexing problem that needs to be addressed to achieve seamless tiled multi-projector displays. In this paper, we present a *scalable real-time* solution to correct the spatial photometric variation in multi-projector displays.

A digital camera is used to *capture* the intensity variation across the display in a scalable fashion. This information is then used to generate a per-pixel *attenuation* and *offset map* for each projector. The former achieves intensity scaling while the later compensates for differing black levels at different regions of the display. These maps are then used to *modify* the input to the projectors to *correct* the photometric variation.

The contributions of our photometric capture and the correction method are the follwing. First, despite the limited resolution of the digital camera, this method is scalable to very high resolution displays. Second, unlike previous methods, this does not require an explicit knowledge of the location of the boundaries of the projectors for generating the attenuation and offset maps. Third, this compensates for the varying black level of the display. Finally, the correction is implemented in real time for OpenGL applications using the distributed rendering framework of Chromium.

## 1 Introduction

Large multi-projector displays are being used extensively for defense, entertainment, collaboration and scientific visualization purposes. Though the issues of geometric seamlessness [21, 20, 24, 8, 19, 6, 5] and scalable driving architecture [12, 9, 2, 10, 11] for such displays have been addressed successfully, we are yet to find a scalable real-time solution for the photometric variation problem.

Methods that use a common lamp for different projectors [18] or manipulate projector controls are labor-intensive, time-consuming and unscalable. Methods that try to match the color gamuts or luminance responses across different projectors by linear transformations of color spaces or luminance responses [15, 22, 23, 3] address only the color variation across different projectors ignoring the variation within a single projector or in the overlap regions. Blending or feathering techniques using software or hardware [21, 14, 4] address only the overlap regions aiming to smooth color transitions across them. However, an adhoc blending function is used instead of deriving it from an accurately measured response of the particular display. Thus, all these methods cannot remove the photometric seams completely. Methods that try to achieve the correction using a digital camera [16] captures the whole display within a single field of view of the camera and can lead to sampling artifacts when used for extremely high resolution displays. Further, the variation in black offset across the display is not compensated.

In this paper, we present a scalable real-time solution for correcting the spatial photometric variation problem. We use a digital camera to capture this variation in a *scalable* fashion. The camera sees only parts of the display at a time and the captured response from these different views are stitched together to generate the photometric response for the whole display. From this response, we generate a per-pixel attenuation and offset map for each projector. The correction for the general photometric variation and the black offset are encoded in these maps. Finally, these maps are used to modify the color inputs to the projectors to achieve the correction. We have implemented this modification in real time using the distributed rendering framework of Chromium.

In Section 2, we describe our scalable algorithm to address the photometric variation in multi-projector displays. The results are presented in Section 3. In Section 4 we discuss some implementation details and the real time implementation on Chromium. Finally, we conclude with future work in Section 5.

## 2 Algorithm

Let us assume that a multi-projector display $D$ is made of $j$ projectors each denoted by $P_j$, $1 \leq j \leq n$. Let

the display coordinates be denoted by $(x, y)$ and the projector coordinates be denoted by $(x_j, y_j)$. Note that at the overlap region, $(x_j, y_j)$ from more than one projector can correspond to a single display coordinate $(x, y)$. Let the sensor used for the measurements be a camera $C$ whose coordinates are denoted by $(x_c, y_c)$. To measure the photometric variation across the display, we need to find the geometric relationship between the display coordinates $(x, y)$, projector coordinates $(x_j, y_j)$ of each projector $P_j$ and the camera coordinates $(x_c, y_c)$. These relationships are given by a geometric calibration method.

**Geometric Calibration:** Let the geometric warp $T_{P_j \to C}$ define the relationship between $(x_j, y_j)$ and the camera coordinates $(x_c, y_c)$ and the warp $T_{C \to D}$ define the relationship between $(x_c, y_c)$ and the display coordinates $(x, y)$. The concatenation of these two warps defines the relationship between the coordinates of the individual projectors and the display, $T_{P_j \to D}$. These warps may be non-linear [8] or linear [20, 24] for different geometric calibration methods.

## 2.1 Capturing the Photometric Variation

It has been shown [17] that for capturing the intensity (photometric) variation of a multi-projector display, we need to find the following.

1. *Projector Channel Intensity Transfer Function :* We need to measure the normalized intensity variation of a channel with increasing input, which is called the *channel intensity transfer function (ITF)*. It has been shown that the channel ITF for projectors may be non-monotonic, but does not vary spatially [17]. Thus, the ITF for each channel needs to be measured at one location for each projector (instead of each pixel). Hence, we call them *projector channel ITF*.

2. *Display Channel White Surface:* We need to measure the maximum intensity that is projected by a channel at every pixel of the display. This per pixel channel maximum intensity defines an intensity surface for each channel which we call the display channel white surface and denote by $W_c$, where $c$ denotes the channel.

3. *Display Black Surface:* We also need to measure the minimum intensity that is present at every pixel of the display when all the different channels are projecting their minimum intensities. This per pixel minimum intensity defines an intensity surface which we call the black surface, and denote by $B$. Basically, this defines the black offset at every pixel of the display.

**Capturing the Projector ITFs:** Previously, we and others have used a point light sensing device (like a spectroradiometer or a photometer) to measure the channel ITF for each projector [16, 17, 15, 24]. However, recent work at University of North Carolina at Chapel Hill uses high dynamic range (HDR) imaging techniques to measure the ITF using only a camera and gets results that are as accurate as those achieved using a spectroradiometer. This makes the method of capturing the photometric variation independent of any expensive sensors like a spectroradiometer or a photometer.

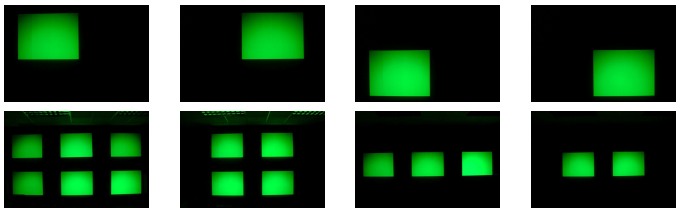

Figure 1: To compute the maximum display luminance surface for green channel, we need only four pictures. Top: Pictures taken for a display made of a $2 \times 2$ array of $4$ projectors. Bottom: The pictures taken for a display made of a $3 \times 5$ array of $15$ projectors.

**Capturing Display Channel White Surface:** We use a digital camera to measure the display channel white surface, $W_c$. There are two known methods to capture $W_c$.

1. In the first [16, 17], the camera is positioned so that it can capture the whole display in its field of view. The camera then captures the image of the maximum projected intensity for each channel of each projector, *one at a time*, when the adjacent overlapping projectors are turned off. The corresponding input that produces this intensity is found from the measured ITF for each projector. A set of such images for the green channel of two different displays are illustrated in Figure 1. The channel white surface for *each projector* is then extracted from the camera image using the warp $T_{P_j \to C}$. These are then added together using the warp $T_{P_j \to D}$. In this method, since the projected image for each projector does not vary greatly in intensity range, a single exposure camera image is sufficient for each of them. If the images from different projectors need different exposures (depending on the relative brightness of different projectors), they are matched in scale using an appropriate scale factor [7].

2. In recent work going on at University of North Carolina at Chapel Hill, HDR images are used to find the display channel white surface. Here also, the camera captures the whole display in its field of view. But instead of capturing images for each projector at a time, the maximum channel intensity for all projectors are projected simultaneously. Then a range of images of

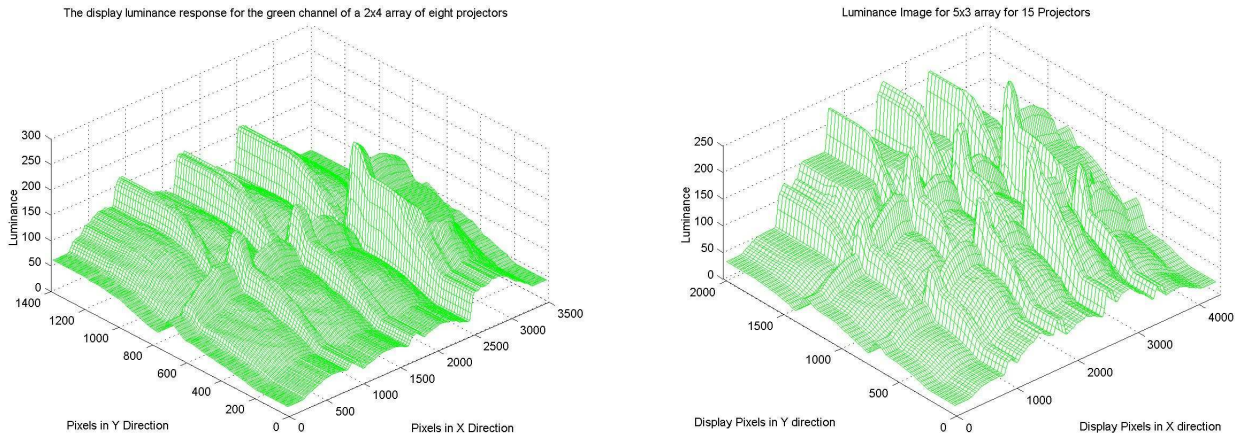


Figure 2: The display channel white surface for the green channel. Left: For a $2 \times 4$ array of eight projectors. Right: For a $3 \times 5$ array of fifteen projectors.

the display is captured by the camera at different exposures. The HDR image generated from these surfaces [7] is used to find the display channel white surface.

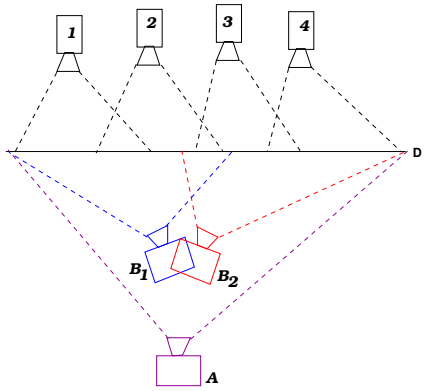


Figure 3: The camera and projector set up for our scalable algorithm.

However, note that both these method suffer from a serious limitation. Since the camera captures the whole display within a single field of view using a limited resolution, this may lead to sampling artifacts for very high resolution displays, like the ones at Sandia National Laboratory (48 projectors) and National Center for Supercomputing at University of Illinois at Urbana Champaign (40 projectors). To remove this limitation, we present a scalable version of the former method. Though we choose the former method, other methods can also be scaled in a similar fashion.

We reconstruct *each projector's* channel white surface using different field of views for different parts of the display and then stitch them together to generate the *display* channel white surface, $W_c$. Figure 3 illustrates the procedure for a display wall made up of four projectors. First, we place the camera at position $A$ when camera's field of view sees the whole display (four projectors) and run a geometric calibration algorithm. The set of geometric warps from this position is denoted by $G_A$.

Next we move the camera to $B$ and/or change the zoom to get higher resolution views of parts of the display. We rotate the camera to see different parts of the display. For example, in Figure 3, the camera sees projectors $P_1$ and $P_2$ from $B_1$ and projectors $P_3$ and $P_4$ from $B_2$. We perform our geometric calibration from these orientations, $B_1$ and $B_2$ to get the corresponding geometric warps $G_{B_1}$ and $G_{B_2}$ respectively.

We also take the pictures for capturing the channel white surface for each projector (similar to Figure 1) from $B_1$ and $B_2$. We reconstruct channel white surface for $P_1$ and $P_2$ from the pictures taken from $B_1$ using $G_{B_1}$, and for $P_3$ and $P_4$ from the pictures taken from $B_2$ using $G_{B_2}$. We stitch these *projector channel white surfaces* using the common geometric warp from $A$, $G_A$, to generate the *display channel white surface*, $W_c$.

The $W_g$ for the green channel thus generated for a $2 \times 4$ array of eight projectors and a $3 \times 5$ array of fifteen projectors are shown in Figure 2. For the former, the display was captured in two parts: the left four projectors and the right four projectors. For the latter, the display was captured in four parts: top left $2 \times 3$ array, bottom left $1 \times 3$ array, top right $2 \times 2$ array and bottom right $1 \times 2$ array.

**Capturing Display Black Surface:** Previous methods have not investigated capturing the black offset that varies at different regions of the display, especially from non-overlap to overlap region. To capture this we again use a digital

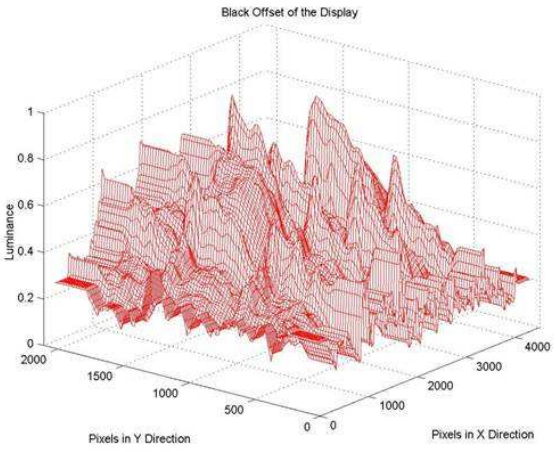


Figure 4: The display black surface for a $3 \times 5$ array of fifteen projectors.

camera and a scalable method exactly similar to that of capturing the display white surface for each channel. The only difference is, in this case each projector projects the minimum intensity from all channels. The corresponding channel inputs are derived from the ITFs. However, note that the exposure used is different. We again use a appropriate scale factor [7] to bring the white and black surface at the same intensity scale. The display black surface, $B$, thus generated for a $3 \times 5$ array of fifteen projectors is shown in Figure 4. Compare the scale of this with the white surface $W_g$ in Figure 2. We found that black surface is about $0.4\%$,$1.25\%$ and $2.5\%$ of the intensity range of green, red and blue channel's white surfaces respectively.

## 2.2 Correcting the Photometric Variation

Now that we have captured the display channel white surface and the display black surface, to achieve photometric uniformity, we need to achieve a *uniform (flat)* response for both of these. Since, the corrected response needs to be within the capabilities (white and black surfaces) of the display, we can achieve the *desired* flat responses by the following.

1. The *desired channel display white surface* should have the intensity which is the *minimum* of the intensities at all pixels of the channel display white surface. Let this desired display white surface be denoted by $W_c'$. Thus,

$$W_c' = \min_{\forall x,y} W_c \tag{1}$$

2. The *desired display black surface* should have the intensity which is the *maximum* of the intensities at all pixels of the display black surface. Let the desired display black surface be denoted by $B'$.

$$B' = \max_{\forall x,y} B \tag{2}$$

In previous work [16], only the former of the above two was done. However, black offset variation is considerable, especially for DLP projectors and needs to be corrected.

It can be shown that $W_c'$ and $B'$ can be achieved by multiplying the input to the display by an *attenuation factor* $S_c(x,y)$, and then adding an *offset* of $O_c(x,y)$ to it, where $S_c(x,y)$ and $O_c(x,y)$ are given by

$$S_c(x,y) = \frac{W_c'(x,y) - B'(x,y)}{W_c(x,y) - B(x,y)} \tag{3}$$

$$O_c(x,y) = \frac{B'(x,y) - B(x,y)}{3(W_c(x,y) - B(x,y))} \tag{4}$$

The $S_c$ and $O_c$ are called the *display attenuation* and *offset maps* respectively. These are then broken up using the geometric warp $T_{P_j \to D}$ to generate the *projector attenuation and offset maps*.

The correction of modifying the input by an attenuation and offset factor assumes a linear projector response. Since this is not true, we need to find the appropriate inputs that would generate the desired channel white and black surfaces when the projector response is non-linear. This is where we use the ITFs for each projector. After modifying the input using the attenuation and the offset factors, we use the inverse ITF on the modified inputs to linearize the projector response. We call the inverse ITF as the *projector channel linearization function*. This linerization function can be represented as a 1D color look-up-table for each channel and is illustrated in Figure 5.

# 3 Results

Figure 6 shows the results of our method. Note that it is not possible to compensate for the black offset perfectly using only software methods. Black offset is the light that is projected by projectors at all times when they are on, even if the input is zero. Usually the leakage light from the projectors is the main contributor to the black offset. Thus, no software attenuation can compensate for it well, since the light cannot be reduced any further by it. The only way to correct it in software is to increase the black offset in the non-overlapping region to match the overlapping regions. This degrades the contrast. However, there are methods that use hardware/optical blending [14, 4] to attenuate light physically and thus can reduce the black offset. Such methods in combination with ours may produce better results.

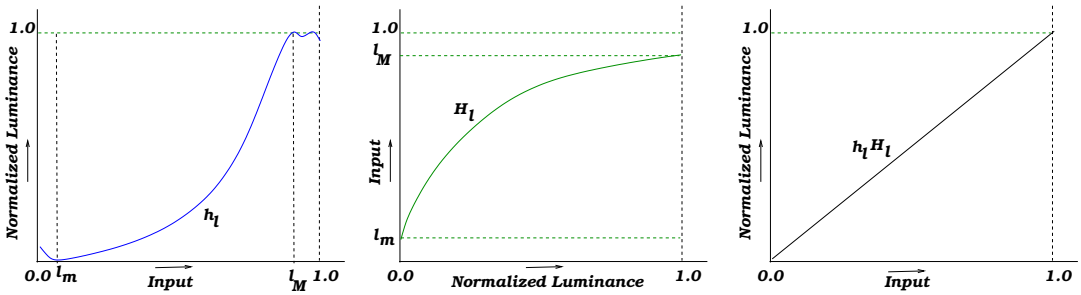


Figure 5: Left: Projector channel intensity transfer function; Middle: Projector channel linearization function; Right: Composition of the channel intensity transfer function and the channel linearization function gives a linear response.

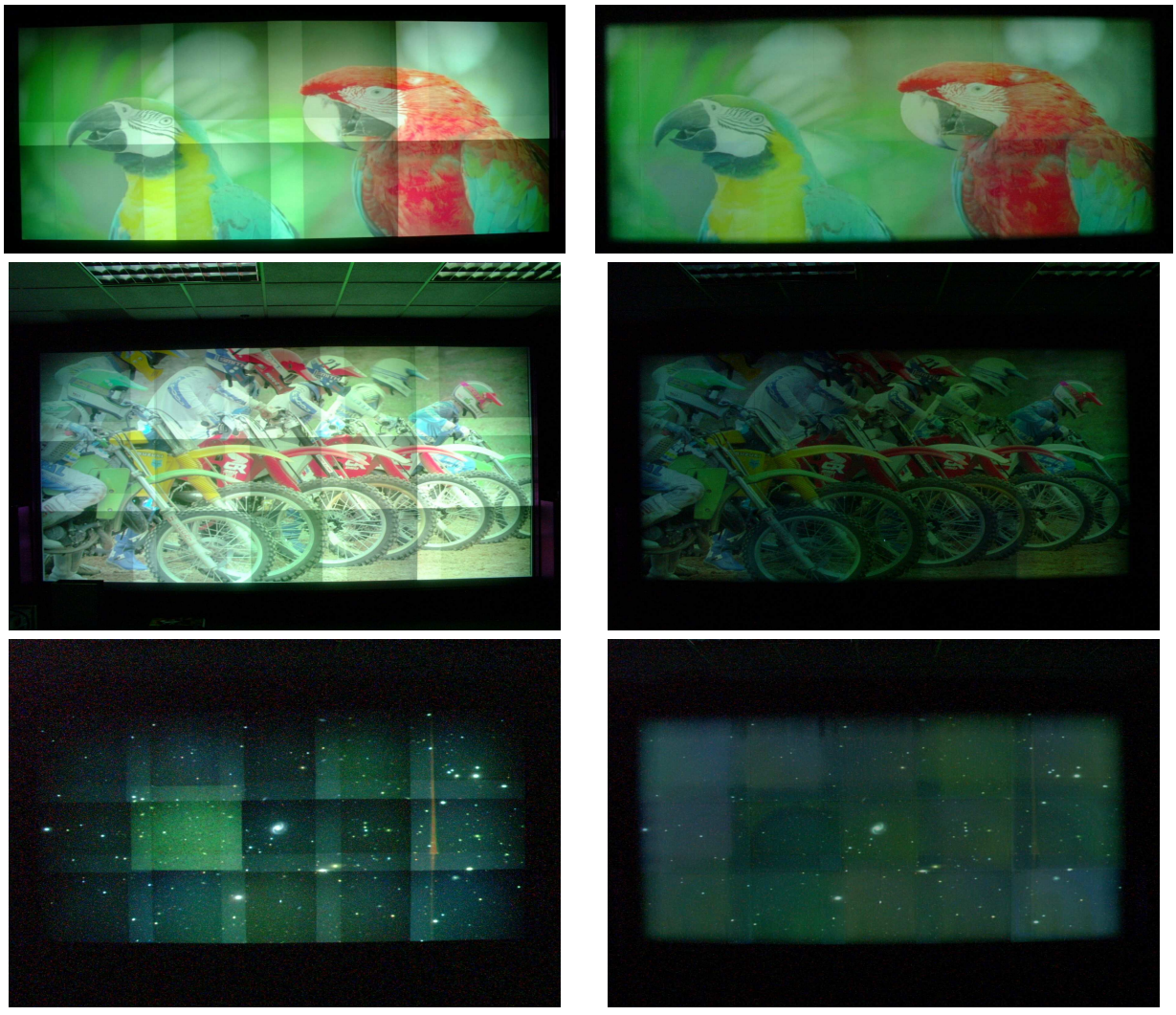

Figure 6: Digital photographs of actual displays. The left column shows images before correction and the right column shows the images after correction. Top: A display with $2 \times 4$ array of eight projector. Our scalable method was used where the left four projectors and the right four projectors were captured separately. Middle: A display with $3 \times 5$ array of fifteen projectors. Here the display was captured in four parts as mentioned in Section 2. Bottom: The same fifteen projector display with the correction shown on a black image of the night sky to illustrate the black correction. These images were taken at a much higher exposure than the other ones.

Another important thing to note is that our correction achieves *photometric uniformity* by achieving identical *linear* response at all display pixels. This makes the display identical to the worst possible pixel (in terms of brightness and contrast) thus ignoring the 'good' pixels which are very much in majority. This leads to severe compression in dynamic range that can be as low as $50\%$ of the original dynamic range of the display.

# 4 Implementation

In this section we discuss a few implementation details that are critical for the success of the algorithm presented in Section 2. Then, we present the real time implementation of the algorithm on Chromium.

## 4.1 Details

**Extracting intensity surfaces from camera RGB image:** A few issues need to be addressed to generate accurate measurements using a camera. To account for the camera's non-linearity, its ITF is reconstructed using the high dynamic range images method presented in [7]. Every camera image used for capturing the display response is first linearized using the inverse of the ITF and then the intensity is extracted using standard RGB to YUV linear transformations [16]. Second, to assure that the camera does not introduce a spatial variation in addition to that which is already present in the display, its aperture is kept below F8. Our experiments and other works [7] show that the spatial intensity variation of the camera is negligible in such setting.

**Removing Noise:** The display white and black surfaces often have noise and outliers due to hot spots in the projectors and/or the camera and the nature of the screen material. The noise and the outliers are removed by a Weiner filter and a median filter respectively. The user provides kernels for these filters by studying the frequency of the outliers and the nature of the noise.

Figure 7: Result with no edge attenuation for a $2 \times 2$ array of four projectors.

**Edge Attenuation:** Usually, the difference in intensity between the non-overlap and the overlap region of the display is very high, thus making the spatial transition very sharp. Theoretically, this sharp transition can only be reconstructed by a camera with resolution at least twice the display resolution. This poses a severe restriction on the process of capturing the photometric response for very high resolution displays. Geometric misregistration of this edge, even by one pixel, creates an edge artifact as shown in Figure 7. To avoid this, we smooth this sharp transition by attenuating a few pixels (40 - 50 pixels) at the edge of the channel white surface and black surface of *each projector* before adding them up the create corresponding surfaces for the whole display. This increases the error tolerance to inaccuracies while capturing the regions of sharp transition. This attenuation is done in software. The attenuation function and width can be controlled by the user. Note that we do not need information about the exact location of the overlap regions for this purpose. Further, this approach allows us to process the intensity surfaces of each projector independently, without explicitly considering geometric correspondences across the projectors. Figure 2 and 4 includes this edge attenuation. To complement this step, we have to put the same edge attenuation back in the projected imagery. This is achieved by introducing it in the the attenuation and offset maps generated for each projector. Empirically, when we use edge blending, even a camera of resolution as low as $25\%$ of the display resolution can produce seamless results with no artifacts.

## 4.2 Real Time Implementation Using Chromium

The algorithm presented in Section 2 generates for each channel of each projector an attenuation map, an offset map and a linearization function. The attenuation and offset maps are per pixel maps of the same resolution as the projector. The linearization function is a 1D color look-up-table for each channel. To correct for the spatial photometric variation, we would need to modify *any* image projected by a projector in the following manner. First, the image is multiplied by the attenuation map. Next, the offset map is added to the resulting attenuated image. Finally, the color of each pixel of the image is mapped to a different color using the linearization function for each channel.

We have developed a small library of functions to apply the corrections to application renderings in real-time. It is implemented to work only with OpenGL and NVIDIA extensions [13], but could easily be extended to work with other extensions and DirectX. The library can be used directly by application programmers. We have also constructed a Chromium [11] SPU enabling OpenGL applications to benefit from the real-time corrections transparently.

| Appl. | O | RT | RTP | RTG | RTPG |
|---|---|---|---|---|---|
| Solar System | 89.3 | 55.5 | 48 | 54.5 | 45 |
| Atlantis | 78.5 | 63.2 | 56.0 | 61.5 | 54 |
| Teapot | 153 | 91 | 72 | 89 | 70 |

Table 1: Performance Analysis (in frames per second) of Photometric Correction Using Chromium on NVIDIA Geforce3 Graphics Cards

Current generation commodity graphics cards with the latest OpenGL extensions are able to provide real-time performance. Our implementation takes advantage of fast render to texture transfers and custom code to drive the texture shaders and register combiners to implement the correction algorithm as follows.

1. The scene rendered as it normally would be, including the application specific use of vertex programs, texture shaders, and register combiners.

2. The framebuffer is then copied using glCopyTexSubImage2D to texture memory for the photometric correction step.

3. The photometric correction is then applied using texture shaders and register combiners. The attenuation and offset maps are applied using multi-texturing. The channel linearization function is then applied by using dependent 2D texture look-ups.

4. The resulting image is then texture mapped to a surface where the geometry correction is applied by manipulation of a mesh of texture coordinates, allowing for a general warping of the image. [24, 8].

5. The final image appears on individual tile of the tiled display with photometry and geometry corrections accounted for.

This approach has been applied to a number of test cases with the result of the various stages of the process highlighted as well as the overall impact on the system. Table 4.2 shows the results (in frames per second) for three different applications, showing the original framerate (O), render to texture (RT), render to texture with photometry correction (RTP), render to texture with geometry correction (RTG), and complete system with render to texture with both the corrections turned on (RTPG).

Table 4.2 shows that it is possible to correct both geometry and photometry errors in a tiled display environment without severe impact on the application's performance. Image warping costs about a millisecond, while photometric corrections cost a few to several milliseconds. Implementing these corrections as a Chromium SPU allows users of this toolkit to benefit from the corrections without having to modify existing applications.

# 5 Conclusion

In this paper, we have presented a scalable method to capture and correct for the spatial photometric variation in multi-projector displays. We have also implemented the correction in real time using the distributed rendering framework of Chromium. This method is being adopted by different educational and research organizations.

However, there are many directions in which a large amount of work needs to be done.

- Our current method produces seamless displays but results in compression of the dynamic range of the display. Several perceptual studies can be investigated to see if we can achieve a *perceptual uniformity* that is less restrictive than strict photometric uniformity, and thus helps us to utilize the system capabilities better. Further, since the linear response for each projector does not cater to the non-linear response of the human eye, the display looks washed out, which needs to be addressed.

- It has been shown [17] that projectors of same model differ significantly photometrically, but are similar in their chrominance or color properties. Hence, for current displays which are usually made of same model projectors, achieving photometric uniformity is often sufficient to give us the desired seamlessness. But, even such displays show visible chrominance variations at certain places that cannot be corrected by any method addressing only photometric variation. On the other hand, methods that address both chrominance and the intensity [22, 23, 1] do not consider the spatial variation. So, new methods need to address all of these together to support more general display configuration. In terms of the real time implementation, it is important to test the system with graphics adaptors that have similar capabilities to those of the NVIDIA adaptors as well as extend it in the lower library to support DirectX.

- Many solutions are emerging that address the general color variation problem. But there is no mathematical framework within which different methods can be evaluated and compared. Such a framework that identifies the key contributing parameters of the color variation problem can also reduce the storage requirements and the complexity of the correction methods.

- Currently we evaluate the results from different algorithms visually, which is subjective. For more objective evaluation, we need a sophisticated perceptual

metric. Such a metric would also help us identify the most important perceptual issues which can then be used as a feedback to improve our algorithms.

## Acknowledgements

The authors would like note contributions to the ideas and experiments leading to the real-time implementation by Gennette Gill, Mark Hereld and Ivan Judson. We acknowledge Kodak for providing us with high-resolution pictures to test our algorithm. This work was supported in part by the U.S. Department of Energy (DOE), under Contract W-31-109-Eng-38.

## References

[1] M. Bern and D. Eppstein. Optimized color gamuts for tiled displays. *ACM Computing Research Repository, cs.CG/0212007, 19th ACM Symposium on Computational Geometry, San Diego*, 2003.

[2] Ian Buck, Greg Humphreys, and Pat Hanrahan. Tracking graphics state for networked rendering. *Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware*, 2000.

[3] Albert Cazes, Gordon Braudaway, Jim Christensen, Mike Cordes, Don DeCain, Alan Lien, Fred Mintzer, and Steve L. Wright. On the color calibration of liquid crystal displays. *SPIE Conference on Display Metrology*, pages 154–161, 1999.

[4] C. J. Chen and Mike Johnson. Fundamentals of scalable high resolution seamlessly tiled projection system. *Proceedings of SPIE Projection Displays VII*, 4294:67–74, 2001.

[5] Han Chen, Rahul Sukthankar, Grant Wallace, and Kai Li. Scalable alignment of large-format multi-projector displays using camera homography trees. *Proceedings of IEEE Visualization*, 2002.

[6] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A.Defanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Proceedings of ACM Siggraph*, 1993.

[7] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. *Proceedings of ACM Siggraph*, pages 369–378, 1997.

[8] Mark Hereld, Ivan R. Judson, and Rick Stevens. Dottytoto: A measurement engine for aligning multi-projector display systems. *Argonne National Laboratory preprint ANL/MCS-P958-0502*, 2002.

[9] Greg Humphreys, Ian Buck, Matthew Eldridge, and Pat Hanrahan. Distributed rendering for scalable displays. *Proceedings of IEEE Supercomputing*, 2000.

[10] Greg Humphreys, Matthew Eldridge, Ian Buck, Gordon Stoll, Matthew Everett, and Pat Hanrahan. Wiregl: A scalable graphics system for clusters. *Proceedings of ACM SIGGRAPH*, 2001.

[11] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahem, Peter Kirchner, and James Klosowski. Chromium : A stream processing framework for interactive rendering on clusters. *ACM Transactions of Graphics*, 2002.

[12] G. Humphreys and P. Hanrahan. A distributed graphics system for large tiled displays. In *Proceedings of IEEE Visualization*, 1999.

[13] M. J. Kilgard. Nvidia opengl extension specifications. 2002.

[14] Kai Li, Han Chen, Yuqun Chen, Douglas W. Clark, Perry Cook, Stefanos Damianakis, Georg Essl, Adam Finkelstein, Thomas Funkhouser, Allison Klein, Zhiyan Liu, Emil Praun, Rudrajit Samanta, Ben Shedd, Jaswinder Pal Singh, George Tzanetakis, and Jiannan Zheng. Early experiences and challenges in building and using a scalable display wall system. *IEEE Computer Graphics and Applications*, 20(4):671–680, 2000.

[15] Aditi Majumder, Zue He, Herman Towles, and Greg Welch. Achieving color uniformity across multi-projector displays. *Proceedings of IEEE Visualization*, 2000.

[16] Aditi Majumder and Rick Stevens. LAM: Luminance attenuation map for photometric uniformity in projection based displays. *Proceedings of ACM Virtual Reality and Software Technology*, 2002.

[17] Aditi Majumder and Rick Stevens. Color nonuniformity in projection-based displays: Analysis and solutions. *Transactions of Visualization and Computer Graphics (to appear)*, 2003.

[18] B. Pailthorpe, N. Bordes, W.P. Bleha, S. Reinsch, and J. Moreland. High-resolution display with uniform illumination. *Proceedings Asia Display IDW*, pages 1295–1298, 2001.

[19] Ramesh Raskar. Immersive planar displays using roughly aligned projectors. In *Proceedings of IEEE Virtual Reality 2000*, 1999.

[20] R. Raskar, M.S. Brown, R. Yang, W. Chen, H. Towles, B. Seales, and H. Fuchs. Multi projector displays using camera based registration. *Proceedings of IEEE Visualization*, 1999.

[21] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image based modeling and spatially immersive display. In *Proceedings of ACM Siggraph*, pages 168–176, 1998.

[22] Maureen C. Stone. Color balancing experimental projection displays. *9th IS&T/SID Color Imaging Conference*, 2001a.

[23] Maureen C. Stone. Color and brightness appearance issues in tiled displays. *IEEE Computer Graphics and Applications*, 2001b.

[24] Ruigang Yang, David Gotz, Justin Hensley, Herman Towles, and Michael S. Brown. Pixelflex: A reconfigurable multi-projector display system. *Proceedings of IEEE Visualization*, 2001.