Vendor-Independent Distortion and Color Correction

John H. Aughey
The Boeing Company
St. Louis, MO
John.H.Aughey@boeing.com

ABSTRACT

The visual display industry has moved away from CRT to fixed-matrix projection technologies. Taking advantage of these high-resolution, low cost projection systems, the military training industry is implementing them in their training devices. In CRT projectors, the geometry errors caused by off-axis projection and other factors can be corrected by shaping the raster; in fixed-matrix projectors this is not possible. In order to align the projected image to a known reference, the rendered image itself must be repositioned and warped so that the viewed image is geometrically correct. Quite often, the resulting image must also be color corrected to accommodate multiple projection sources or screen materials.

The distortion of the image and color correction can be applied in the image generator, in the projector, or anywhere in between. As fixed-matrix projection systems appear to be the dominant technology in the industry for the foreseeable future, establishing a vendor-independent standard for image distortion and color correction that has strictly defined behaviors and is repeatable across all projection technologies and image generator vendors will greatly facilitate development of products using these technologies.

In this paper, I will provide a baseline set of requirements necessary to support fixed-matrix projection technologies for both flat and curved screens. From these requirements I will derive a set of needs within the image pipeline, from image generation to the illuminated projection surface, to achieve the desired image shape. Finally, I will discuss how these distortion functions can be applied to the image, both as relates to position and color, and propose a verifiable standard for how the application of these functions affects the final projections. Additionally, I define interface needs for man-in-the-loop calibration and maintenance within established protocols such as CIGI.

ABOUT THE AUTHOR

John Aughey is a software engineer at The Boeing Company with over 14 years of visual display experience. John has developed visual systems for Integrated Defense Systems, including F/A-18 simulator projects and Internal Research and Development activities. John is a co-inventor on two visual-systems related patents. John received his Bachelors degree in Computer Science from Purdue University and a Masters degree in Computer Science from Washington University in St. Louis.

Vendor-Independent Distortion and Color Correction

John H. Aughey
The Boeing Company
St. Louis, MO
John.H.Aughey@boeing.com

INTRODUCTION

The training industry is developing more complex visual systems every year. Through the use of low cost, high resolution projectors, and the innovative use of faceted, curved, and domed screen surfaces, affordable and highly immersive training solutions are becoming the norm rather than the exception.

Projector technologies fall into two general categories – CRT (Cathode Ray Tube) and Fixed-Matrix. CRT projectors are few and far between these days. The projection industry has embraced fixed-matrix projection technologies such as LCD (liquid crystal display), DLP™ (Digital Light Processing), and LCOS (liquid crystal on silicon) due to their low manufacturing and operation cost. These projectors have been used in low fidelity simulators, as well as high resolution projectors with excellent light output levels meeting requirements for very high fidelity simulators.

Issues of Alignment

On nearly every projection system, the projected image must be adjusted to fit within the screen surface. Even after the image is optically aligned using focus, zoom, and lens offset, the image often needs further refinement to achieve the desired shape. What follows is a look at how the projected image can be manipulated beyond the limits of the optical system.

Fixed-matrix projectors can be generalized by the conceptual diagram in figure 1.

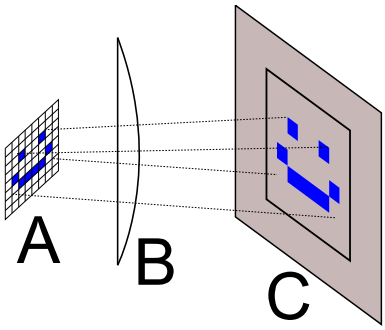


Figure 1 - Fixed Matrix Generalization

Pixels, generated by a computer, are displayed on a physical object plane (A). Conceptually, light emits from each of these pixels, is spread by a lens (B), and ultimately illuminates a surface (C). LCD, DLP, and LCOS projection technologies carry out this process in different ways, but the important part to note is a pixel from (A) will always project to a single point on (C). The optics in (B) can modify the pixel spread and even provide some nominal geometric distortion such as fish-eye lens effects. The principle of pixel distribution of (A) will be distributed on (C) within the physical properties of the optics in (B).

Without additional alignment controls, the ability to position a pixel from (A) to a location (C) is limited by the physical (and expensive) optics in (B). CRT projectors have more dynamic control over the physical pixel positions because the electron beam sweeps over a continuous phosphor, allowing the position of the beam to manipulate pixels. Fixed-matrix projectors, on the other hand, do not have this control because the illuminated pixels are not movable. In practice, the optics are specified to achieve the best fit when projected, but this best fit is not likely to meet alignment requirements. Additional calibration is always necessary in order to meet the specifications of the visual system.

Distortion of the image

Conference room style projectors have alignment controls such as size, offset, and keystone to adjust for typical off-axis projection effects. More complex controls such as barrel distortion and zone adjustments are found on more expensive projectors giving even more precise control of the image.

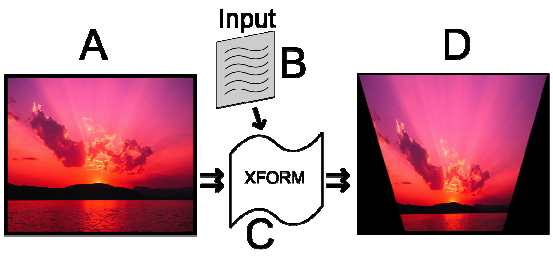


Figure 2 - Keystone Correction

In order to move a pixel projected on (Figure 1 C), the information contained in that pixel must be moved to another pixel before it is displayed on the object plane (Figure 1 A). Figure 2 shows a conceptual process that transforms one 2-D image into a second 2-D image with some modification. The transformation shown is a simple keystone adjustment intended to flatten a non-orthogonal projection.

The transformation process shown in part C of Figure 2 is the focus of this paper. The paper will define the content of the input (B) into the process. It will define how the process (C) shall modify the input image (A) to produce the output image (D) given the input (B). Finally, this paper will discuss ways of communicating the transformation data (B), along with methods to validate that the entire process is performing according to a known standard.

GOALS

The goal of this paper is to establish a mathematical basis for distortion and color correction. This paper does not define the actual methods of implementation to perform distortion/color correction, but rather specifies the information that goes into a distortion/color correction system and how an image processor should act on that data.

This is intended to be the opening discussion of a formal standard for distortion/color correction. Whereas a formal standard will define specific byte structures and messaging protocols, this paper will bring to light the elements that will ultimately be defined using a formal standard process. The ideas presented below are not all inclusive, and it is expected that major players in image generation, projector design, and image analysis will collaborate to form a coherent formal standard.

TARGET AUDIENCE

Correcting for projection distortion is not limited to the projector. Often, distortion correction is performed within the image generator itself. The process shown in figure 2 can be performed in many different places. Consider a typical display pipeline shown in figure 3.

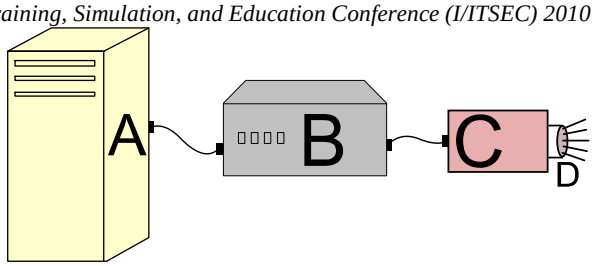


Figure 3 - Display Path

At each stage in this display path, the rendered image can be modified. In figure 3: within the Image Generator (IG) (A), techniques such a multi-pass rendering can distort the output image. Between the IG and the projector (B), third-party distortion correction hardware can be used to perform the necessary distortion. As mentioned above, projectors (C) have built-in capabilities to modify the image. And finally, images can be dynamically distorted in optics (D). The definitions and specifications in this paper are appropriate for defining distortion correction at any stage in the display path.

PROBLEM DOMAIN

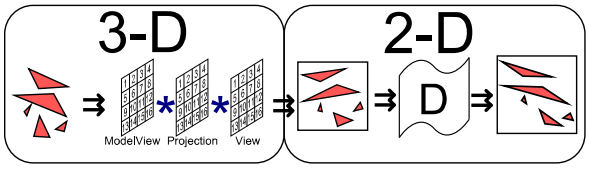


Figure 4 - Rendering Pipeline

The figure above shows a simplified pipeline for rendering a 3-D scene. Triangles are transformed by a variety of model-view matrices. These triangles are then transformed by a projection matrix and finally normalized to a rectangular 2-D viewing surface. Once rendered to 2-D image, this image can be further distorted and modified to fit the physical projection space.

This paper considers is what comes after the 3-D pipeline. The definitions that follow consider in the 2-D image space only. It is possible for a clever implementation to obtain the same results entirely within a 3-D rendering pipeline, but in order to define the distortion correction process, only the image transformation done in a 2-D rectangular space is used.

As described in the TARGET AUDIENCE section, defining the distortion entirely within the 2-D image space widens the reach of this proposed specification. If this specification is supported by all possible implementation domains, this well-defined method of

performing image distortion can be leveraged by simulation, entertainment, scientific research, home theater, and many other use cases.

DEFINITIONS

A 2-D object is an entity that can be addressed using an x,y coordinate. This object returns a value when addressed using a valid x,y coordinate and may also be writable. An embodiment may be represented as a two-dimensional array of values. Alternatively, it could be a continuous mathematical function *f(x,y)* with no inherent internal size.

Coordinate System

The coordinate system definition for a 2-D object defines the axis system and addressable limits of the object within that axis system. Because only the 2-D space is under consideration, only one coordinate system will be defined. Figure 5 illustrates this coordinate system.

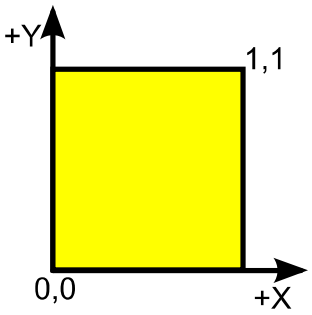


Figure 5 - 2-D Coordinate System

All 2-D objects will be mapped to this [0,1] unit coordinate system. (0,0) addresses the extreme lower left value while (1,1) addresses the extreme upper right value. Note that this coordinate system is an inclusive interval where (0,0) and (1,1) are accessible addresses. Addresses less than (0,0) or greater than (1,1) are not defined unless otherwise specified.

Addressing 2-D objects using a floating point unit space allows objects be of any size (e.g., 2x2, 4x16, 46x153). All 2-D objects are addressable using this unit space regardless of their actual size. The implementation is free to internally re-map this unit space into a different vector space as needed, but for the purposes of providing an interface and definition, this unit space is used.

Lookup Functions

2-D objects are addressable using the coordinate system defined above. At each addressable point, there is a value that can be obtained. For 2-D images, this value might be a {red, green, blue} tuple. For other 2-D objects, the value might be a single value or multiple values depending on the type of data contained within. No matter the type of value returned, the nomenclature used to denote a value lookup is:

OBJECTNAME(x,y)

Where *OBJECTNAME* is the 2-D object that is being addressed, and *(x,y)* is the address within the unit space coordinate system.

Continuous Value Space

All 2-D objects are continuous throughout their addressable value space. This means that any valid address within the 2-D object will contain a valid value.

Figure 6

Because the coordinate system does not have discrete integer values, an address might not fall on a pixel or element boundary. Consider the 2x2 image in figure 6 above. The address (0,0) is solidly red. The address (1,1) is solidly green. However, the address (0.5,0.5) refers to the point in the middle of the image that does not fall on a discrete pixel boundary. Is the value obtained by IMAGE(0.5,0.5) RED, GREEN or some value in-between?

It is expected that an implementation will provide, at a minimum, linear interpolation when looking up an in-between value within a 2-D object. For the above figure, IMAGE(0.5,0.5) would return YELLOW.

DISTORTION MAPS

The INTRODUCTION used the example of a projector correcting for a keystone distortion effect as an adjustable parameter within the projector setup. It is often the case that the external user-interface controls are simplified by using well known terms. For a typical conference room projector, keystone, size, shift, and image flipping adjustments do a decent job to make an acceptably shaped image. But these projectors usually do not inherently allow the image to be adjusted for fish-eye lenses or spherical distortion effects when projecting onto a domed surface.

The number of projector controls and definitions needed to align any projected image to any surface is infinite. It is not practical to mathematically define every conceivable distortion function for every possible configuration. To allow for any conceivable image distortion, a 2-D distortion map is used.

In figure 2, the input into the image transformation function is this 2-D distortion map. Multiple maps may be used depending on how the image is to be transformed both in positional space and color space.

Distortion Map Definition

A 2-D distortion map is a 2-D object, defined using the same guidelines in the DEFINITIONS section above. A distortion map can be as large or as small as necessary. A given implementation that uses these techniques may have *a priori* knowledge of the actual projector and rendering resolutions in order to achieve desired alignment accuracy, but this specification does not give a minimum or maximum size of any 2-D object.

At this point, only the existence of a rectangular distortion map is known. The sections that follow explain how this distortion map is used to transform one 2-D image into another.

Forward and Reverse Transforms

Two types of transformations are available to transform the input image to the output image. Each transformation can obtain an equivalent end result, but the methods are different.

Forward Transform

The forward transform takes a value from the source image, and writes that value to an address in the destination image using the distortion map to determine the destination image address. An embodiment of this technique is commonly used where the scene is first rendered to a texture and that texture is then rendered a second time on a distorted polygon mesh.

Reverse Transform

The reverse transform is just the opposite. The reverse transform considers an address in the destination image, and reads a value from the source image using the distortion map to determine the source image address.

Reasons for choosing one over another

A particular implementation may choose one transformation over another because of internal addressing constraints or other performance considerations. For example, the Forward Transform may require writes to an address in a 2-D object to be done at a sub-pixel resolution. These writes need to occur without adverse results such as 'holes' appearing in the resulting image where pixels were not specifically addressed. When the 3-D rendering pipeline is used in a two-pass render, the texture rasterization done on the GPU (Graphics Processing Unit) will not have unintended side effects. But other implementations that do not have full rendering capabilities may not be able to perform a forward transform.

Transforming a forward distortion map to a reverse distortion map is not trivial, so a complete system should compute a forward or reverse distortion map that the implementation supports.

Distortion Function

Given all the definitions above, the actual distortion function is quite simple for both the forward and reverse transforms. The functions below show the work that occurs within an image processor that performs geometric distortion.

In these functions, *DEST* is the destination 2-D image, *SOURCE* is the source 2-D image, and *MAP* is the distortion map.

Forward Transform

For each pixel address (x,y) in source image
DEST(MAP(x,y)) = SOURCE(x,y)

Reverse Transform

For each pixel address (x,y) in destination image
DEST(x,y) = SOURCE(MAP(x,y))

For each of these transforms, *MAP* is used as a lookup function to convert one *x,y* address to another. In the forward transform, the value returned by *MAP* is the new destination address. In the reverse transform, it is the source address.

If *MAP(x,y)* returns a value that cannot be addressed (the value is outside of [0,1]), then the image lookup will return a black color value.

EXAMPLE TRANSFORMATIONS

Using the distortion map definition above, simple transformations can be done with extremely low resolution distortions maps, while complex transformations can be achieved at any accuracy

requirement by using larger size distortion maps. Below are examples of transformations.

Identity Transformation

An important transformation is one where the source and destination images are equivalent. In fact, for any distortion map size greater than 2, there is an identity map.

Smallest Identity Map

0,1	1,1
0,0	1,0

Figure 7

Above is the smallest identity map. Using the coordinate definitions, address (0,0) will return the lower left value which is (0,0). Similarly, address (1,1) will return the upper right value which is (1,1). The linear interpolations addressing requirements return all values in-between so (0.5,0.5) returns the average mixture of (0,0) and (1,1) which is (0.5,0.5).

This identity map shows *MAP(x,y) == (x,y)* so in both the forward and reverse lookup functions, *DEST(x,y) = SOURCE(x,y)* resulting in equivalent source and destination images.

Similar Identity Maps

Distortion maps of any size have unique identity maps. To create an identity map, the lower left value is (0,0) and each subsequent value linearly transitions to 1 along each axis. A generalization of an identity map for a large distortion map is shown in figure 8.

0,1	…	0.5,1	…	1,1
…	…	…	…	…
0,0.5	…	0.5,0.5	…	1,0.5
…	…	…	…	…
0,0	…	0.5,0	…	1,0

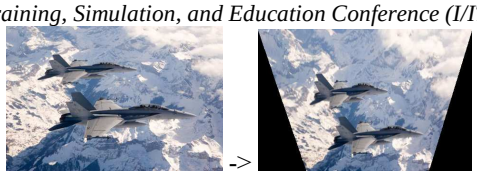
Figure 8

Simple Distortion Maps

For illustration, some simple distortion maps using forward transforms are illustrated below.

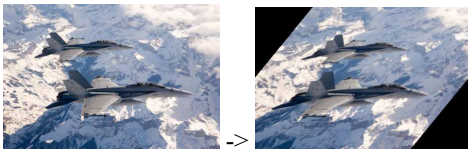
Keystone

0,1	1,1
0.25,0	0.75,0



Shear

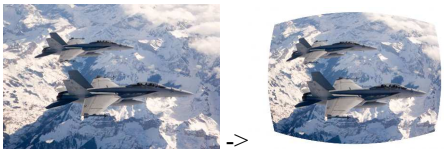
0.25,1	1.25,1
-0.25,0	0.75,0



Complex Distortion Maps

A higher resolution map can perform more complex distortion functions. In domed display systems, the image often requires non-linear reshaping to correct for distortion effects produced by viewing the rectangular projection on a spherical surface. At the extreme, a distortion map with dimensions equal in size to the rendered images could be used to move each and every pixel precisely. In practice, much lower resolution maps can be used, taking advantage of the linear interpolation of both the images and the distortion lookup maps.

Below is an illustration of a lens distortion using a 32x32 distortion map.



COLOR CORRECTION

Color correction in projection systems is necessary for a variety of reasons. When multiple projectors are used together in a single visual system, a color from one projector must match the color of another projector so that the image appears to be continuous without distracting changes in color. Often a projection system must be tuned so that a projected color is at a specific color coordinate when measured with a chromaticity meter. In blended projection systems as shown in Figure 9, two adjacent projectors must seamlessly fade from one projector to another. The intensity of one projected image must fall off at the same rate that the intensity of the second image increases.

*SOURCE(MAP(x,y)) * ATTENUATION(x,y)*

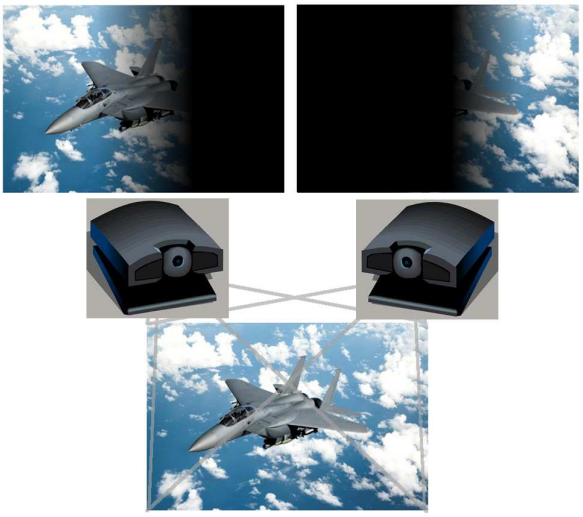


Figure 9 – Blending two projectors

The computation necessary for a color correction function is highly variable depending on the color manipulation required to achieve the desired effect. Intensity attenuation could be a simple multiplier, but even this may require more complex mathematics to account for non-linear intensity profiles such as projector gamma. Color-space manipulation can become very complex, requiring unique coefficients depending on a pixel's position within the image.

The specific implementation needs to define the data necessary to perform the color correction required for that visual system. However, the same map definitions described above can be used to supply the necessary information to the implementation-specific processing.

Intensity Attenuation

To illustrate a basic color correction function, consider the requirement to attenuate the image by some percentage depending on the pixel location. This sort of intensity attenuation could be used for projection masking (attenuating 100% outside the screen surface) or for simple blending between two projectors.

Using the Reverse Transform method,

For each pixel address (x,y) in destination image
DEST(x,y) = SOURCE(MAP(x,y))

Becomes

For each pixel address (x,y) in destination image
DEST(x,y) =

More complex color conversion functions may use multiple maps, or provide more complex mathematics within the manipulation code, but the format and communication of the data to the processor can leverage the standards already defined.

VERIFICATION

For any requirement or standard, a method of verification is necessary. Implementations of distortion and color correction are no exception.

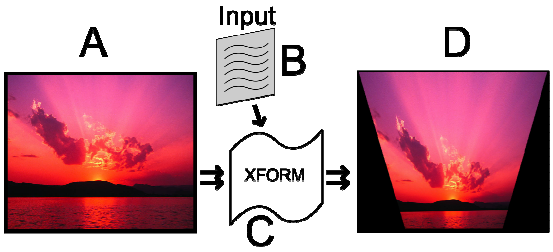


Figure 10 – Transformation Process

Needs

The process that needs verification is shown above in figure 10. To verify this process, the following steps are performed:

1. Supply a distortion map (B)
2. Supply an image to be distorted (A).
3. Apply the distortion to the image (C).
4. Read the resulting distorted image (D).
5. Verify the results against expected behavior.

Steps 1, 2 and 4 require support from the image processor. The exact method steps 1, 2 and 4 use is implementation specific, and depends highly on the physical interface of the particular device. Computer based image generators would be expected to conform to a defined communications protocol, while projectors or in-the-loop distortion hardware may have other means of performing these steps.

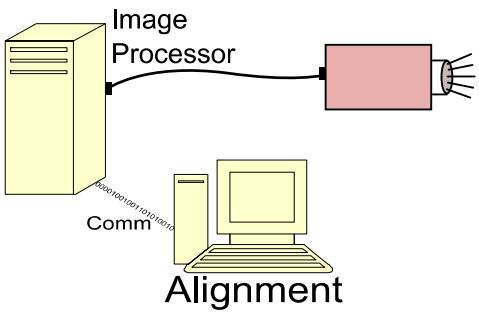
Equivalency

The word equivalent is used here to describe two images that are the "same". An exact bit-for-bit copy of one image to another is equivalent by definition. An image that is "almost" identical might be equivalent depending on an external specification.

The coordinate system definition implies that a computation occurs to convert an integer pixel space to a continuous unit space. Additionally, the linear interpolation adds an additional layer of inexactness to the process. In theory, there is no loss of precision, but in practice information is lost when performing floating point computations.

The verification procedure must account for possible inexactness in the image transformation. Program requirements may specify a tolerance for allowable error. The desired error is 0, but the error should be no more than ½ pixel. In practice, the error should be much less than ½ pixel, but some degree of imprecision should be expected. Any verification procedure should expect error, and be able to quantify that error.

COMMUNICATION



For alignment and operation, data must be transmitted between an alignment or configuration management system and the image processor itself. At a minimum, the distortion/color correction maps must be communicated to the image processor. Ideally, changes in distortion/color maps can be sent and applied in real-time allowing for interactivity with an alignment system.

Leveraging Existing Protocols

When communicating with image processors, existing protocols should be leveraged when appropriate in order to re-use established communications channels. Care should be taken so that communicating map data using an existing protocol does not make the implementation overly complicated.

Protocol Requirements

Some high level requirements can be derived without defining specific byte level messaging protocols. A fundamental piece of data that must be sent is the distortion/color correction map itself.

Looking at how the distortion function is applied, a distortion map is an *n* by *m* array of floating point number pairs. The minimum size map required to do anything practical is 2 by 2 requiring a total of 2*2*2=8 values. A larger map of 32 by 32 would require 2048 values. A map that distorts each pixel of a standard HD image requires over 4 million values. If single precision floating point values were used as the data type, these maps would require 32, 8192, and 16,588,800 bytes respectively just for distortion data alone. What is clear from this back-of-the-envelope calculation is inherent support within the protocol to send reliable variable length bulk data.

Common Image Generator Interface

The Common Image Generator Interface (CIGI) is a protocol going through the Simulation Interoperability Standards Organization (SISO) standardization process. It is used as a communications protocol between a host computer and an image generator to communicate view frustums, eye point angles, entity information, and other data for military simulation visualization. For a visual simulation product, CIGI is a natural protocol to consider extending to communicate distortion data to an image generator computer.

CIGI uses UDP which is by definition an unreliable protocol. This is appropriate for entity information that changes often, but inappropriate for stateful information that is only sent once. IG acknowledge messages can help to verify reception of data, but unreliable communication is only one piece of the problem.

CIGI is non-streaming, meaning that data is sent in small chunks with no inherent ordering. The protocol is designed for very small messages such as entity positions. Within a CIGI message, the current packet size is limited to 255 bytes. If a double-precision floating point map is sent using this packet size, only 14 points could be sent per packet – accommodating only the smallest of all possible distortion maps. UDP then limits a single message to 64k. The chunking techniques necessary to support larger maps, along with creating a new reliable acknowledgment protocol make sending large amounts of data over the existing CIGI standard difficult. It is clear that the existing CIGI protocol is not the appropriate transfer mechanism for bulk data.

It is possible to extend CIGI to support bulk data transfer, and various proposals have been presented to the SISO committee to incorporate bulk data transfers within the CIGI protocol. One proposal uses TCP as the transport layer for reliable streaming communication

and adding a bulk data message to the CIGI specification. The specifics of the implementation are beyond this paper, but should be explored as this matures.

BENEFITS

Once a standard is established, a statement such as, "X product supports Y standard" answers an entire domain of possible interface questions. As systems are designed and proposed using any number of third party solutions, the knowledge that a product verifiably supports a standard improves the accuracy of proposals and reduces costs overall. If system A requires distortion correction to be performed by component B, this standard can be used by B to confirm that requirement, or implement the necessary capability without additional coordination.

In addition to reducing end-product costs, product development and research is improved by being able to test and interchange many different available solutions without complex re-work. Companies can independently implement their solution against this standard or even re-use existing implementations.

From an engineering point of view, there is a great benefit in having common behavior among display products. It is important that a distortion map will distort an image equivalently, pixel for pixel, when given to different distortion correction implementations. During the development of a curved screen product, a man-in-the-loop alignment system was created that provided real-time interactivity during the calibration process. This system ran independently from the image generator. Once calibrated, the distortion map was sent to the image generator for use in the simulation. A common map format and distortion behavior insured that the alignment generated by the calibration system would be accurate pixel for pixel in the run-time system.

NEXT STEPS

As discussed above in the GOALS section, this paper is an open invitation for other players to collaborate to form an industry standard for distortion and color correction. An established standard benefits all players, reducing costs throughout. The information in this paper was derived from a small fragment of the known visual systems, and additional capability needs are requested.

ACKNOWLEDGEMENTS

I would like to acknowledge The Boeing Company for their support of this paper. Additionally, Aechelon Technology, Inc. provided assistance for the integration of these ideas into a commercial product. Finally, I acknowledge the I/ITSEC Simulation Subcommittee for their support and assistance throughout this process.

REFERENCES

Dan Brockway, Stephen Gersuk, Elizabeth Smith. Non-Linear Distortion Correction. *SigGraph, 2008, Course 20*

Rob Lechner, Willard, Phelps. CIGI, a Common Image Generator Interface. *I/ITSEC, 2002*

Remi Arnaud et al. Multi-purpose high resolution distortion correction. *USPTO 6249289, Jun 19, 2001*

Stephanie S. Chen et al. Correction for keystone distortion in a digital image displayed by a digital projector. *USPTO 6491400, Dec 10, 2002*