# Non-Uniform Crosstalk Reduction for Dynamic Scenes

F.A. Smit*
Center for Mathematics and Computer Science

R. van Liere†
Center for Mathematics and Computer Science

B. Froehlich‡
Bauhaus-Universität Weimar

**ABSTRACT**

Stereo displays suffer from crosstalk, an effect that reduces or even inhibits the viewer's ability to correctly perceive depth. Previous work on software crosstalk reduction focussed on the preprocessing of static scenes which are viewed from a fixed viewpoint. However, in virtual environments scenes are dynamic, and are viewed from various viewpoints in real-time on large display areas.

In this paper, three methods are introduced for reducing crosstalk in virtual environments. A non-uniform crosstalk model is described, which can be used to accurately reduce crosstalk on large display areas. In addition, a novel temporal algorithm is used to address the problems that occur when reducing crosstalk in dynamic scenes. This way, high-frequency jitter caused by the erroneous assumption of static scenes can be eliminated. Finally, a perception based metric is developed that allows us to quantify crosstalk. We provide a detailed description of the methods, discuss their tradeoffs, and compare their performance with existing crosstalk reduction methods.

**Keywords:** Crosstalk, Ghosting, Stereoscopic display

**Index Terms:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms

## 1 INTRODUCTION

Stereoscopic display systems allow the user to see three dimensional images in virtual environments. For active stereo with CRT monitors, Liquid Crystal Shutter (LCS) glasses are used in combination with a frame sequential display of left and right images. When the left image is displayed the right eye cell of the LCS glasses goes opaque and the left eye cell becomes clear, and vice versa.

Stereo systems suffer from a disturbing effect called crosstalk or ghosting. Crosstalk occurs when one eye receives a stimulus which was intended for the other eye. This produces a visible shadow on the image that reduces, or even inhibits, the viewer's ability to correctly perceive depth. The effect is most noticeable at high contrast boundaries with large disparities.

Three main sources of crosstalk can be identified: phosphor afterglow, LCS leakage and LCS timing [9]. Typical phosphors used in CRT monitors do not extinguish immediately after excitation by the electron beam, but decay slowly over time. Therefore, some of the pixel intensity in one video frame may still be visible in the subsequent video frame. Also, LCS glasses do not go completely opaque when occluding an eye, but still allow a small percentage of light to leak through. Finally, due to inexact timing and non-instantaneous switching of the LCS glasses, one eye may perceive some of the light intended for the other eye. The combination of these effects causes an undesirable increase in intensity in the image, which is visible as crosstalk. The amount of crosstalk increases drastically towards the bottom of the display area in a non-linear fashion. Hence, crosstalk is non-uniform over the display area.

To enhance depth perception we want to eliminate or reduce the effect of crosstalk. One way to achieve this is by using better, specialized hardware, such as display devices with faster phosphors and higher quality LCS glasses. However, this type of hardware is relatively expensive and might not eliminate crosstalk completely. An alternative solution is to reduce the effect of crosstalk in software by processing and adjusting the image frames that are to be displayed.

The governing idea is to subtract an amount of intensity from each pixel in the displayed image to compensate for the leakage of intensity from the previous video frame. A pre-condition here is that the displayed pixels have enough initial intensity to subtract from. If this is not the case, the overall intensity of the image has to be artificially increased, thereby loosing contrast. As such, there appears to be a tradeoff between loss of contrast and the amount of crosstalk reduction possible.

Previous work on software crosstalk reduction mostly focussed on the preprocessing of static scenes with a fixed viewpoint [4] [5]. However, what is needed is crosstalk reduction in virtual environments; i.e. for dynamic scenes, various viewpoints and in real-time. Since large screens with different viewpoints are used, crosstalk can no longer be assumed to be uniform over the entire display area. Furthermore, as virtual environments operate in real-time, crosstalk reduction algorithms must be fast enough to run in real-time.

After having implemented an initial version of the crosstalk reduction algorithm, we discovered some high frequency jitter for fast moving objects. This effect was not visible when crosstalk reduction was disabled. No such problem has previously been mentioned in the literature. After further investigation, we found this to be a defect in the crosstalk reduction algorithm due to the assumption of static scenes. All previous crosstalk reduction methods correct a single pair of left and right application frames, which is valid only for static scenes. For dynamic scenes the application frames change slightly over time, and are displayed in sequence with preceding application frames. Therefore, crosstalk induced from previous application frames needs to be considered. Upon closer inspection this turns out to be a problem that is related to the video refresh rate (e.g. 100 Hz) being different than the application frame rate (e.g. 20 Hz). In this case there is no longer a one-to-one correspondence between video frames and application frames.

In this paper, our contribution is threefold:

- A non-uniform crosstalk model, along with a procedure for interactive user calibration of the model parameters. The method is based on a combination of the non-uniform crosstalk characteristics described by Woods and Tan [9] and the calibration procedure proposed by Konrad et al. [4]. This addresses the problem of non-uniform crosstalk over the display area.

- A real-time, temporal crosstalk reduction algorithm, which addresses the problem of changing frames in dynamic scenes. We developed an efficient approach to consider the temporal high-frequency jitter, and were able to completely remove it. The resulting algorithm runs entirely on the GPU and is fast enough to operate in real-time.

*e-mail: Ferdi.Smit@cwi.nl
†e-mail: Robert.van.Liere@cwi.nl
‡e-mail: bernd.froehlich@medien.uni-weimar.de

- A quantitative evaluation methodology for the perceptual quality of crosstalk reduction. Such a method is useful for the development and comparison of various crosstalk reduction techniques. The proposed evaluation method measures the amount of perceptually disturbing crosstalk, allowing us to make quantitative assessments about the quality of various crosstalk reduction algorithms.

The methods in this paper have been applied to the Personal Space Station [7], a desktop virtual environment that consists of a head tracked user behind a 22 inch CRT monitor using active stereo. However, we will show that these methods can be beneficial to other virtual environments, in particular those environments with large display areas.

## 2 Related Work

Lipscomb and Wooten [5] described a method to reduce crosstalk in software. The background intensity is artificially increased, after which crosstalk is reduced by decreasing pixel intensities according to a specifically constructed function. The screen is divided into 16 horizontal bands, and the amount of crosstalk reduction is adjusted for each band to account for the non-uniformity of crosstalk over the screen. Also, some additional adjustments are made in the case of thin lines. Although a non-uniform model is used, the difficulty with this method is determining proper function parameters that provide maximum crosstalk reduction for each of the 16 discrete bands. We propose a non-uniform model that is continuous and can be interactively calibrated.

A calibration based method was proposed by Konrad et al. [4]. First, the amount of crosstalk caused by an unintended stimulus on a prespecified intended stimulus is measured by a psychovisual user experiment. The viewer has to match two rectangular regions in the center of the screen in color. One contains crosstalk and the other does not. After matching the color, the actual amount of crosstalk can be determined. The procedure is repeated for several values of intended and unintended stimuli, resulting in a two dimensional look-up table. This look-up table is inverted in a preprocessing stage, after which crosstalk can be reduced by decreasing pixel intensities according to the inverted table values. Optionally, pixel intensities are artificially increased by a contrast reducing mapping to allow for greater possibility of crosstalk reduction. A drawback of this method is that it assumes crosstalk is uniform over the height of the screen. Our method uses a similar calibration procedure, but is based on a non-uniform crosstalk model.

Klimenko et al. [3] implemented real-time crosstalk reduction for passive stereo systems. Three separate layers are combined using hardware texture blending functions. The first layer contains the unmodified left or right image frame to be rendered. The second layer is a simple intensity increasing, additive layer to allow for subsequent subtraction. Finally, the left image is rendered onto the right image as a subtractive alpha layer and vice versa. The alpha values are constant but different for each color channel. This is a linearized version of the subtractive model of Lipscomb and Wooten [5]. Although the method works in real-time, it does not take into account the interdependencies between subsequent image frames. We have improved this with a temporal model. Also, with constant alpha values the model is uniform over the screen, and some effort is needed to determine the proper alpha values. Our method is continuous non-uniform and can be interactively calibrated. Futhermore, the model and implementation are completely linear due to hardware restrictions. By using modern GPU fragment programs and frame buffer objects we avoid such restrictions, and are able to implement a non-linear correction model completely on the GPU.

Woods and Tan [9] studied the various causes and characteristics of crosstalk. They showed that most CRT display devices use phosphors with very similar characteristics, such as spectral response and decay times. However, there was a considerable amount of variation in the quality of LCS glasses. The resulting crosstalk model shows that the amount of crosstalk due to phosphor afterglow increases heavily towards the bottom-end of the screen. Crosstalk due to leakage of the LCS glasses is almost uniform over the first 80% of the display, but slightly reduces in the bottom 20%. Finally, to evaluate the real amount of crosstalk, photographs of the screen were taken through the LCS glasses using a digital camera. Our quantitative evaluation method to determine the quality of crosstalk reduction is based on this approach. Also, we examined the validity of our non-uniform model by comparing it with the described crosstalk characteristics.

Daly [1] proposed the Visible Differences Predictor (VDP) to estimate the perceptual difference between two still images. Two images are taken as input, and an output image containing the probability of perceivable difference for each pixel is produced. The algorithm operates using a frequency domain weighting with the human contrast sensitivity function, followed by a series of detection mechanisms based on the human visual system. We will compare digital photographs taken through the LCS glasses in this manner to estimate the perceptual quality of crosstalk reduction methods.

## 3 Methods

In this section we will give a detailed technical description of our methods. First, we will describe the non-uniform crosstalk model and its calibration procedure in Section 3.1. Next, we describe how to use the calibration data for the implementation of crosstalk reduction for temporal scenes in Section 3.2. Finally, our quantitative evaluation method is described in Section 3.3.

### 3.1 Crosstalk Model and Calibration

The amount of perceived crosstalk can vary depending on a number of factors, such as lighting, monitor brightness and different hardware. Most crosstalk reduction methods, for example the method described by Lipscomb and Wooten [5], need to be adjusted for varying conditions. However, it might not be immediately clear to the users exactly how to do this. Therefore, a simple way of calibrating the crosstalk reduction algorithm is desirable.

To estimate the amount of crosstalk correction required in a given situation, we make use of a calibration method based on a similar psychovisual experiment by Konrad et al. [4]. The latter experiment assumes crosstalk to be uniform across the screen. However, as was shown by Woods and Tan [9], this is not the case and results in poor crosstalk reduction towards the bottom of the screen. What is needed is a crosstalk model, combined with an easy to use calibration method, that can handle the non-uniform nature of crosstalk.

#### 3.1.1 Non-uniform Crosstalk Model

We first describe the non-uniform nature of crosstalk using a simple function of screen height $y$. This function need not be exact; it only serves as an indication of the shape of the non-uniformity for crosstalk reduction purposes. We separate the crosstalk into crosstalk due to leakage of the LCS glasses, and crosstalk due to phosphor afterglow. The former can almost be approximated as being uniform over screen height, while the latter exhibits a strong non-linear shape [9]. To approximate this shape, we modeled the typical decay of CRT phosphors using a power-law decay function [2]

$$\phi(t,i,y) = i * (t - y + 1)^{-\gamma}$$

where $t \in [0,\infty)$ stands for the time after the first vertical blank (VBL) signal, $i$ is the intensity of the phosphor in the next video frame (at $t = 1$), $y \in [0,1]$ is the screen height, and $\gamma$ is a constant indicating the speed of decay. As the amount of crosstalk is constant over a specific scanline, $\phi(t,i,y)$ disregards the pixel column $x$ and

| | Left | Right |
|---|---|---|
| Left | $I_{intended}$ | $I_{adjust}$ |
| Right | $I_{unintended}$ | $I_0$ |

| | Left | Right |
|---|---|---|
| Left | $I_{desired}$ | $I_{adjust}$ |
| Right | $I_0$ | $I_{unintended}$ |

Table 1: (Left) Calibration setup for a uniform crosstalk model. The columns show the two halves of the screen, while the rows show the sequentially displayed left and right eye frames. By altering $I_{adjust}$ the amount of crosstalk between $I_{intended}$ and $I_{unintended}$ can be determined. (Right) An alternate calibration setup for our non-uniform model. By altering $I_{adjust}$ we find the intensity to display, given $I_{unintended}$, such that the user perceives $I_{desired}$

only depends on $y$. It is assumed that the first frame is displayed for $t \in [0,1)$, the second for $t \in [1,2)$, etc.

To estimate the perceived intensity of the phosphor in the second frame, we integrate $\phi(t,i,y)$ over $t$:

$$c(i,y) = \int_{1.05}^{2.0} \phi(t,i,y)\,dt = i \cdot \left( \frac{1}{\gamma(2.05-y)^\gamma} - \frac{1}{\gamma(3-y)^\gamma} \right)$$

We integrate over $t \in [1.05,2]$ because the electron beam spends a small amount of time in the vertical retrace (VRT) state, at which time the LCS glasses are still opaque. We have also experimented with exponential decay functions, but found that a power-law decay function better approximates reality. Still, $c(i,y)$ is only an approximation of shape, and does not model the physical reality exactly.

Suppose that we fix $y$ to the screen height of a given pixel we wish to correct. We can now vary two parameters, $\gamma$, and $i$ to estimate the amount of crosstalk. Also, a constant term needs to be added, which is a combination of a constant amount of crosstalk for the phosphor decay and for the LCS glasses. This leads to the following calibration function

$$ct(\sigma,\varepsilon,\gamma) = \sigma + \varepsilon \cdot \left( \frac{1}{\gamma(2.05-y)^\gamma} - \frac{1}{\gamma(3-y)^\gamma} \right)$$

Note, when $\gamma$ is fixed, the rightmost term can be precalculated as it only depends on $y$. The value for $\sigma$ describes the amount of crosstalk at the top of the screen, $\varepsilon$ at the bottom, and $\gamma$ is a parameter indicating the curvature of the function. The three parameters can be set to closely match the shape of the crosstalk curves found by Woods and Tan [9].

### 3.1.2 Uniform Crosstalk Calibration

To calibrate the display we use a procedure similar to the one used by Konrad et al. [4]. We shall first describe Konrad's method in more detail, after which we describe the changes required for our non-uniform model.

First the screen is divided into a left and a right half. At each screen half a certain constant intensity is displayed for the right and left eye in stereo mode, as shown in Table 1. As stated earlier, only the center of the screen is used here, and crosstalk is assumed to be uniform. When the screen is viewed through LCS glasses where the left eye is shuttering normally and the right eye is always opaque, the following can be seen on the screen:

$$\begin{aligned} Left: &\quad I_{intended} + \theta(I_{unintended}, I_{intended}) \\ Right: &\quad I_{adjust} + \theta(I_0, I_{adjust}) = I_{adjust} \end{aligned}$$

where $\theta(u,i)$ is a function describing the amount of crosstalk between an intended intensity $i$ and an unintended intensity $u$ in the other eye. When $u = 0$ it is assumed there is no crosstalk, so $\theta(0,i) = i$. When the users matches the two screen halves in intensity by adjusting $I_{adjust}$, this results in the equality: $I_{intended} + \theta(I_{unintended}, I_{intended}) = I_{adjust}$. Hence, the amount of crosstalk between an intended and unintended intensity can be specified as:

$$\theta(I_{unintended}, I_{intended}) = I_{adjust} - I_{intended}$$

The procedure is repeated for several values of $I_{intended}$ and $I_{unintended}$, and for each red, green and blue color channel. This results in three look-up tables, one for each channel, that map from $\mathbb{R}^2 \to \mathbb{R}$. Next, these tables are linearly interpolated and inverted by an iterative procedure to be able to correct for crosstalk. This results in a function that maps a given unintended and desired perceived intensity to the intensity to be displayed:

$$\theta^{-1}(I_{unintended}, I_{desired}) = I_{displayed}$$

### 3.1.3 Non-uniform Crosstalk Calibration

In this section we shall describe the changes required to calibrate the non-uniform crosstalk model. The parameter space of our non-uniform model is multi-dimensional, as opposed to one dimensional. Our calibration procedure will return function parameters $\sigma, \varepsilon, \gamma$ for a function depending on the screen height $y$ of the pixel. Therefore, interpolating and inverting the resulting calibration tables is not a straightforward task. To avoid this problem, we change the calibration setup slightly as shown in Table 1. Instead of determining the amount of crosstalk an unintended intensity produces on an intended intensity, we directly estimate the intensity to be displayed, given a desired intensity and an unintended intensity. This is equivalent to calibrating for $\theta^{-1}$.

To match both screen halves in intensity, the user first adjusts $\sigma$ in order to match the top part of the screen. Then, $\varepsilon$ is adjusted to match the bottom part of the screen. Finally, $\gamma$ is adjusted for the proper curvature. The user adjustable intensity is set to:

$$I_{adjust} = 1 - \min(1, ct(\sigma,\varepsilon,\gamma))$$

This is an approximation of the inverse of $ct(\sigma,\varepsilon,\gamma)$ up to a constant. Hence, when displaying $ct^{-1}$, the crosstalk is canceled out. The approximation results in slightly too little crosstalk correction towards the bottom of the screen; for values of $y$ close to 1. However, as was shown by Woods and Tan [9], the LCS glasses produce slightly less crosstalk at the bottom of the screen. We previously estimated this LCS leakage to be constant, so the small error we make by estimating the inversion compensates for this assumption.

When the two halves match, the following can be seen:

$$\begin{aligned} Left: &\quad I_{desired} + \theta(I_0, I_{desired}) = I_{desired} \\ Right: &\quad I_{adjust} + \theta(I_{unintended}, I_{adjust}) = \\ &\quad 1 - \min(1, ct(\sigma,\varepsilon,\gamma)) + \theta(I_{unintended}, I_{adjust}) \end{aligned}$$

If the amount of crosstalk has been modeled correctly, that is $\theta \approx ct$, the values of $I_{adjust}$ and $\theta(I_{unintended}, I_{adjust})$ will compensate each other up to the constant $I_{desired}$. In other words, given an undesired intensity, we know what intensity to display for the user to perceive the desired intensity. Note, $I_{adjust}$ occurs both by itself and in $\theta(I_{unintended}, I_{adjust})$, therefore we interactively solve a complex equality. Also, all equations depend on the height $y$ of the current pixel.

User guided calibration of three correction parameters, on a two-dimensional grid of intended and unintended parameters, for each color channel, is a tedious and time-consuming process. After some initial experimentation, we found that the value for $\gamma$ could be fixed at 6.5 in all cases on our hardware. We expect this to be true for most hardware, as $\gamma$ depends on the type of phosphors used, and CRT monitors are known to use very similar phosphors [9]. This reduces the parameter space to only $\sigma$ and $\varepsilon$. To interpolate the calibration grid for uncalibrated values of $I_{desired}$ and $I_{unintended}$ we used linear interpolation on the remaining function parameters $\sigma$ and $\varepsilon$.

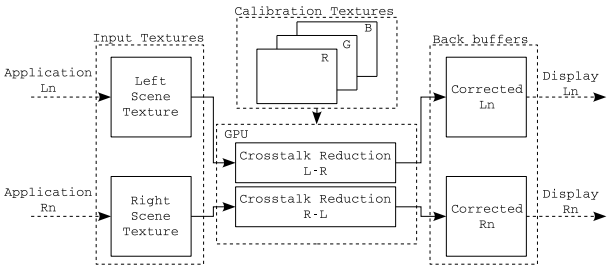


Figure 1: A simple crosstalk reduction pipeline for static scenes. Left and right scene textures are taken as input and are processed on the GPU using the calibration table textures. The output is written directly to the left and right back buffers using MRT-2.

### 3.2 Application of Temporal Crosstalk Reduction

In this section we describe the implementation of the temporal crosstalk reduction algorithm on the GPU. First, we will describe the simple case of crosstalk reduction, assuming a static scene. Then the required changes for dynamic scenes are discussed.

#### 3.2.1 Crosstalk Reduction for Static Scenes

The algorithm starts out by rendering the stereo scene for the left and right eye to two separate textures. Next, crosstalk reduction is performed, and finally the resulting left and right textures are displayed as screen sized quads. The only requirements are modern video hardware, capable of running fragment programs, and an existing pipeline which is able to render to textures using frame buffer objects (FBOs). An overview of the static crosstalk reduction pipeline is given in Figure 1.

Once the scene has been rendered to the left and right floating point textures, they are bound as input to the reduction algorithm. Note that texture targets are bound to video hardware buffers, therefore everything runs entirely on the GPU; no texture downloads by the CPU are required. The calibration grid is interpolated and stored in three separate 256x256 input textures for each of the red, green and blue calibration colors. This results in three calibration textures, $T_{Red}$, $T_{Green}$ and $T_{Blue}$. The textures have a 32 bit floating point, 4-channel RGBA format.

The calibration textures map the pair of pixel intensities $(I_{unintended}, I_{desired})$ to the two parameters $\sigma$ and $\varepsilon$ of the calibration function (see Section 3.1). As this only requires two of the four available texture channels, we also store the inverted pair of intensities as an optimization. For $c \in \{Red, Green, Blue\}$, the calibration textures have the following layout:

$$T_c(u,v) \rightarrow (\sigma^c_{u,v}, \varepsilon^c_{u,v}, \sigma^c_{v,u}, \varepsilon^c_{v,u})$$

The left and right scene textures, as well as the three calibration textures, are then passed to a hardware fragment program by drawing a screen sized quad. In this way, screen pixels can be processed in parallel on the GPU. When drawing a pixel intended for the left application frame, $I_{desired}$ is fetched from the left scene texture and $I_{unintended}$ from the right scene texture, and vice versa. Next, the corresponding values of $\sigma^c$ and $\varepsilon^c$ are fetched from the calibration textures for all three color channels.

From the fetched calibration parameters, and the assumed constant $\gamma = 6.5$, a correction function depending on the screen height $y$ of the pixel can be constructed. We perform crosstalk reduction by setting:

$$I^c_{displayed} = 1 - \min(1, \sigma^c_{u,v} + \varepsilon^c_{u,v} \cdot f(y))$$

where $c \in \{Red, Green, Blue\}$ as before, $u = I^c_{unintended}$, $v = I^c_{desired}$ and

$$f(y) = \left( \frac{1}{\gamma(2.05-y)^\gamma} - \frac{1}{\gamma(3-y)^\gamma} \right)$$

Note, as $f(y)$ does not depend on either $\sigma$ or $\varepsilon$, it needs to be calculated only once for a fixed value of $y$. Also, all of the above calculations can be done in the fragment program.

Instead of rendering the crosstalk corrected left and right application frames separately in two passes, they can be rendered in a single pass using hardware Multiple Render Targets (MRT). This way, the left and right back buffers are bound simultaneously for writing.

In the dual pass case, correcting a single pixel for either the left or right application frame takes five texture lookups: two to fetch the desired and unintended pixel intensities from the left and right scene textures, and three to fetch the corresponding calibration parameters. This results in a total of ten texture lookups when rendering the left and right application frames.

However, when using MRT both sets of calibration parameters can be fetched simultaneously due to the structure of the calibration textures. Only five texture lookups are required in this case: two to fetch the pixel intensities from the left and right scene textures, and three to fetch the calibration parameters for $(I^c_{left}, I^c_{right})$ and $(I^c_{right}, I^c_{left})$ simultaneously from $T_c$. Also, both corrected pixels can be written to their corresponding left or right back buffer immediately.

It may appear that we make an error because we do not compensate the crosstalk reduction for the crosstalk reduction itself in the previous frame; i.e. it seems we should compensate the left frame $n+1$ with respect to the corrected right frame $n$ and not to the right frame $n+1$. However, for static scenes this is not the case, as our calibration procedure already determines the proper intensity to be displayed given a desired perceived intensity. As the calibration procedure performs a similar correction itself, the correct parameters are found implicitly.

An inherent problem with subtractive crosstalk correction methods is that one cannot compensate the crosstalk when the desired intensity is too low compared to the unintended intensity. When the crosstalk estimation for a color channel results in a value higher than the desired intensity for that channel, the best one can do is to set the channel to zero. In order to avoid this problem, the desired image intensity can be artificially increased. One way of doing this is by the following linear mapping for $\alpha \in [0,1]$: $I_{desired} = \alpha + I_{desired}(1-\alpha)$. We optionally allow this kind of intensity increase in our implementation. However, a big drawback of increasing intensity this way is that it significantly reduces contrast.

#### 3.2.2 Crosstalk Reduction for Dynamic Scenes

So far we have assumed static scenes in the application of crosstalk reduction. However, in virtual environments this is not the case. In an interactive application, subsequent application frames are usually slightly different. For frame sequential, active stereo the application frames are displayed sequentially as $L^a_n, R^a_n, L^a_{n+1}, R^a_{n+1}, ...$, where $L^a$ and $R^a$ stand for the left and right application frames respectively. The problem lies in the fact that $R^a_n$ and $R^a_{n+1}$ are slightly different due to animation. The static crosstalk reduction algorithm combines $L^a_{n+1}$ with $R^a_{n+1}$ to produce a corrected left frame, while it should combine $L^a_{n+1}$ with $R^a_n$. This results in incorrect crosstalk reduction at the differing regions of $R^a_n$ and $R^a_{n+1}$.

The problem is complicated by the fact that rendering an application frame usually takes much longer than the display of an individual video frame. At a monitor refresh rate of 100 Hz in active stereo mode, ie. 50 Hz per eye, application frames need to be rendered in under approximately 5 ms per eye. Typical applications can not maintain this frame rate, and thus per application frame, many sequential video frames are displayed. The situation is sketched in

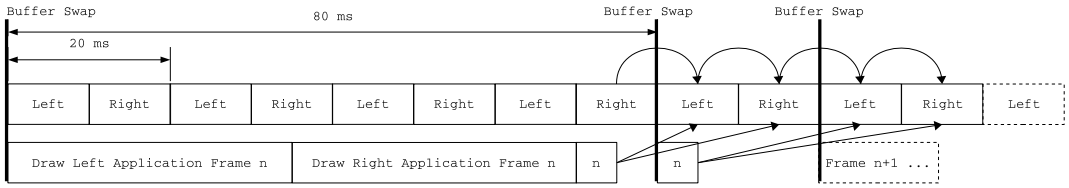


Figure 2: Time sequence diagram of video and application frames in dynamic scenes. A refresh rate of 100Hz is assumed here. During the time it takes to render the left and right application frame N, eight left and right video frames are displayed. At the first buffer swap, the last video right frame causes crosstalk on the first video left frame after the bufferswap. These two video frames belong to different application frames N-1 and N. The curved arrows on top indicate the pairs of video frames our algorithm corrects between. The straight arrows at the bottom show which frames are rendered when. At the second buffer swap the right and left video frames belong to the same application frame N. This figure shows the unoptimized case, where all four video frames are corrected to the previous corrected video frame.

Figure 2. As crosstalk only exists between video frames, the slow rendering of application frames needs to be taken into account as well.

When application frames can be rendered at the same speed as video frames, the crosstalk reduction algorithm can be easily modified to take dynamic scenes into account. In addition to the left and right scene textures $S^L_{n+1}$ and $S^R_{n+1}$ for application frame $n+1$, the corrected right output $C^R_n$ for application frame $n$ is used as input as well. Now $S^L_{n+1}$ is corrected with respect to $C^R_n$, resulting in $C^L_{n+1}$. Next, $S^R_{n+1}$ is corrected with respect to $C^L_{n+1}$, resulting in $C^R_{n+1}$. The procedure can now be repeated for the next application frame $n+2$.

When application frames can not be rendered as fast as video frames, the procedure is no longer correct. Suppose that during the time to render a left and right application frame, four left and four right video frames are displayed. In this case, the above algorithm corrects $S^L_n$ with respect to $C^R_{n-1}$. However, this is only correct for the first time this video frame is displayed. The same left application frame will be displayed three more times, and those times it should have been corrected with respect to $C^R_n$.

When rendering an application frame takes much longer than the display of a video frame, we can not properly correct every video frame on a single videocard. This is due to the fact that video hardware executes rendering commands in a sequential manner. At some point the application frame's drawing commands need to be executed by the hardware, at which time there is no way to correct the video frames displayed during this period. An alternative solution would be the use of two videocards: one for rendering application frames, and one for correcting every video frame. Assuming that the transmission and correction of frames can be done fast enough, this will allow the crosstalk to be reduced correctly for every video frame.

Our implementation partially addresses this problem by drawing ahead two pairs of left and right images for the same application frame, which are displayed as subsequent video frames. In this way we correct between new video frames, and the previously corrected video frame: $S^L_{n+1} - C^R_n$, $S^R_{n+1} - C^L_{n+1}$, $S^L_{n+1} - C^R_{n+1}$ and $S^R_{n+1} - C^L_{n+1}$. However, as more than four video frames are usually displayed per application frame, the last two corrected video frames will be repeated multiple times. Also, for example, $C^R_{n+1}$ is not very different from $S^R_{n+1}$. Therefore, we optimize this procedure and use the following correction sequence: $C^L_n = S^L_n - C^R_{n-1}$, $C^R_n = S^R_n - S^L_n$, $C^L_{n+1} = S^L_n - S^R_n$, and $C^R_{n+1} = S^R_n - S^L_n$

This allows us to use the optimization to correct $S^L_n$ and $S^R_n$ both ways using five texture lookups as discussed in the previous section. Also, $C^R_{n+1}$ can be set to this same result, after which it is used as input to the next application frame. The four corrected frames can be drawn to four textures in a single pass using MRT-4, after which the first two textures can be displayed, followed by the last two textures after a buffer swap. This is shown in Figure 3.

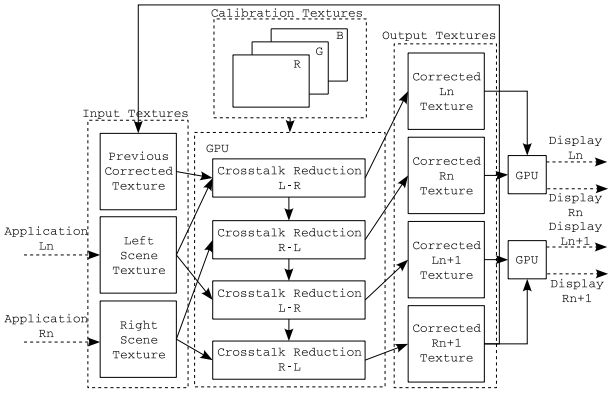


Figure 3: Crosstalk reduction pipeline for dynamic scenes. Again the left and right scene textures are taken as input, together with the output of the previous run. Crosstalk reduction is then performed on the GPU, and the output is written to four textures using MRT-4. The first pair is then displayed, followed by a buffer swap, after which the second pair is displayed. The display sequence conforms to Figure 2. This figure shows the unoptimized case. In the optimized case the correction methods directly use the scene textures, instead of the output of the previous correction.

### 3.3 Evaluation Method

To evaluate the quality of different crosstalk reduction algorithms, we developed a way to quantitatively compare the amount of crosstalk reduction. One of the problems is the fact that crosstalk is only visible to an external viewer and can not be detected in the rendering system itself. Therefore, some way of registering the real amount of generated crosstalk is required for both corrected and uncorrected scenes.

Another important issue is that we only wish to measure the amount of crosstalk that is perceptually disturbing. Crosstalk increases the overall intensity of the display in a non-uniform way; however, when the intensity increase is gradual this is not perceptually disturbing or even discernable. Two images can differ a lot in their pixel values without showing any perceptually discernable differences. Therefore, statistical methods of comparison, such as the root mean squared error (RMSE), do not provide us with any useable information. This discrepancy of statistically based comparison methods was noted before by various authors (an overview is given by McNamara [6]).

To externally register the amount of crosstalk in a given scene, we take digital photographs of the monitor screen through the LCS glasses. The digital camera is placed in front of the left eye of the LCS glasses, which are shuttering in the normal frame synchro-

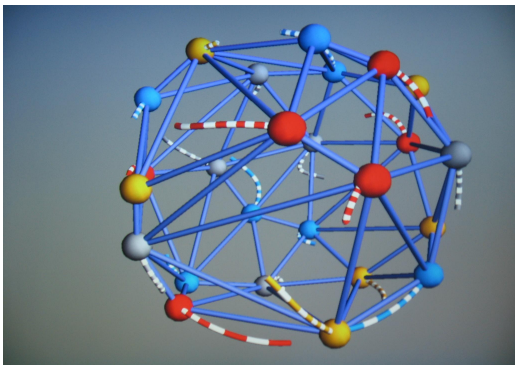

Figure 4: A photograph of the reference frame where no crosstalk is present. The photo was taken through the left eye of activated LCS glasses.

nized manner. This setup is similar to the one used by Woods and Tan [9].

Using this setup, we take a photograph of a reference application frame $P_{ref}$, where the left scene is displayed in the left eye and the right eye display is kept blank. In this way no crosstalk is present in the reference frame. A sample reference photograph is shown in Figure 4. Next, we take a photograph $P_{ct}$ of the same application frame, where the left and right scenes are displayed as normal in the left and right eyes. This photo includes crosstalk for the left scene, originating from the right scene. Finally, we take another photo $P_{cor}$ in the same manner, but this time with the crosstalk reduction algorithm enabled. All photographs are taken with identical camera settings and a relatively slow shutter speed.

The crosstalk in $P_{ct}$ and $P_{cor}$ can be isolated by comparing these photographs to $P_{ref}$, which is clear of crosstalk. When no crosstalk is present, the left eye image should be equal to $P_{ref}$. To evaluate the perceptually disturbing differences between $(P_{ref}, P_{ct})$ and $(P_{ref}, P_{cor})$ we make use of the Visible Differences Predictor (VDP) by Daly [1].

To estimate the perceptual differences between two images, the VDP first applies a non-linear response function to each image to estimate the relation between brightness sensation and luminance. Next, the image is converted into the frequency domain and weighted by the human contrast sensitivity function (CSF), resulting in local contrast information. Additional sub-band processing based on the human visual system (HVS), in combination with masking functions, provides scaled contrast differences between the two images. Finally, these contrast differences are used as input to a psychometric function to determine the probability of perceptual difference. More specific implementational details of the method are described by Daly [1].

The output of the VDP algorithm is a probability map that indicates for each pixel the probability a human viewer will perceive a difference between the corresponding two pixels in the input images. In order to quantify the results, we determine the percentage of pixels that are different with a probability of over 95%. The threshold was liberally chosen because the typical VDP output per pixel was either very high (>95%), or very low (<5%), with hardly any values in between. This provides us with a measure of the amount of perceptually disturbing crosstalk.

Crosstalk increases the intensity of the entire display area, as even the background causes crosstalk on itself. However, these gradual differences in intensity over larger areas do not cause perceptually disturbing artifacts. For the crosstalk photos compared to the reference photos, the RMSE per scanline at the bottom of the screen is very large as the crosstalk photo is much brighter there. However, the VDP output shows this does not cause a disturbing or even perceivable difference. In this way, by using a human perception based comparison, we only quantify the truly disturbing visible differences between the crosstalk and reference images.

## 4 Results

In this section we describe experimental results, and compare our crosstalk reduction algorithm to the case where no crosstalk reduction is performed, and to Konrad's uniform reduction model [4]. To evaluate the amount of disturbing crosstalk we use the evaluation method described in Section 3.3.

NuVision LCS glasses were fixed in front of a Canon A520 digital camera to take photographs of an Iiyama Vision Master Pro 512 22 inch CRT screen. The monitor displayed a frame by frame animation of the optimization procedure for globally optimal Fekete point configurations [8]. Each application frame was displayed in reference mode ($P_{ref}$), with normal crosstalk ($P_{ct}$), with non-uniform crosstalk reduction enabled ($P_{cor}$), and with Konrad's uniform crosstalk reduction ($P_{uni}$) in sequence. This allowed us to take photographs of all application frames, for each of the display modes. The setup was previously described in Section 3.3.

In this manner we took photographs of 800 animation frames, resulting in a total of 3200 digital color photos of 2048x1536 resolution. The photos were taken with the following fixed camera settings: shutter speed 0.5s, aperture F2.8, fixed manual focus, fixed white balance, ISO 50, with the highest quality settings. The camera was operated automatically over USB in a darkened room, ensuring frame synchronization and unchanged conditions.

For each animation frame, we compared each of the photos $P_{cor}$, $P_{ct}$ and $P_{uni}$ to the corresponding reference frame $P_{ref}$ using the previously described evaluation method. Figure 5 shows the three photos and the output of the VDP comparison with the reference photo for animation frame 570. The reference photo for frame 570 was shown earlier in Figure 4. The percentage of perceptually different pixels for each frame and correction method has been plotted in Figure 6.

From Figure 6 it can be seen that both uniform and non-uniform crosstalk reduction provide significantly better results than no crosstalk reduction. Also, non-uniform reduction is significantly better than uniform reduction; especially when the animation progresses. This can be explained by the fact that when the animation has been running for some time, more geometry becomes visible at the bottom of the screen. The parameters of the uniform reduction model have been calibrated only for a specific location on the display, in this case the center. The amount of crosstalk becomes larger at the bottom of the screen and the uniform model does not correct this. However, the non-uniform model takes this into account and corrects for crosstalk everywhere. This can also be seen in the photos of frame 570 in Figure 5.

When the desired pixel intensity to be displayed is very low, and the corresponding unintended intensity due to crosstalk is fairly high, no adequate crosstalk reduction can be performed. We distinguish between two types of crosstalk: *object-to-background* and *object-to-object*. Object-to-background crosstalk is caused by a geometric object and is visible on the background. This is generally the most perceptually disturbing kind of crosstalk. Object-to-object crosstalk is the crosstalk that is visible on the surface of an object, often caused by the object itself.

In most cases object-to-background crosstalk can be corrected, as the animation background intensity of 0.6 for all three color channels is fairly high. However, object-to-object crosstalk could not be corrected in all cases. This is due to the fact that the object often does not have a high enough desired intensity in a specific color channel to reduce the crosstalk caused in that channel, for example when a blue object region causes blue crosstalk on a red object region. In this case the desired intensity of the blue channel is zero, while the unintended intensity is not.

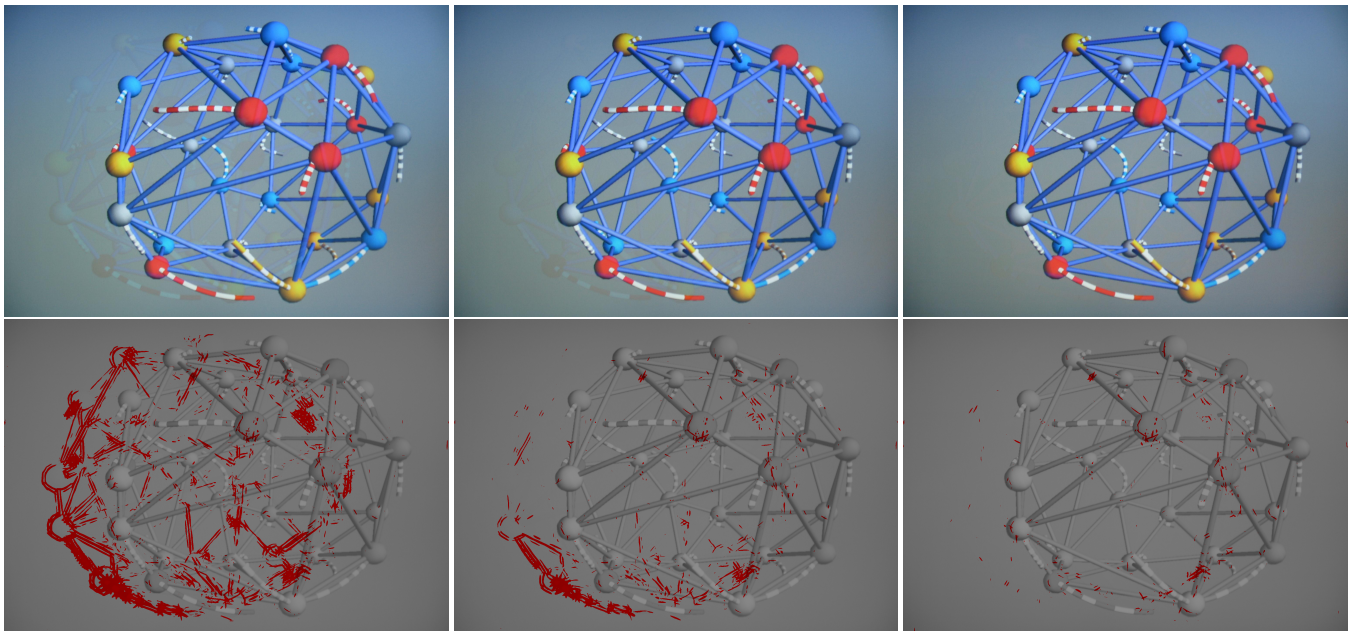

Figure 5: Photographs and evaluation for frame 570 of the animation. The top row shows the photos as captured by the camera. The bottom row shows the corresponding perceptual differences with the reference photo in red. From left to right the images represent: no crosstalk reduction, uniform crosstalk reduction, and non-uniform crosstalk reduction. Similar MPEG videos of the entire animation sequence are available.

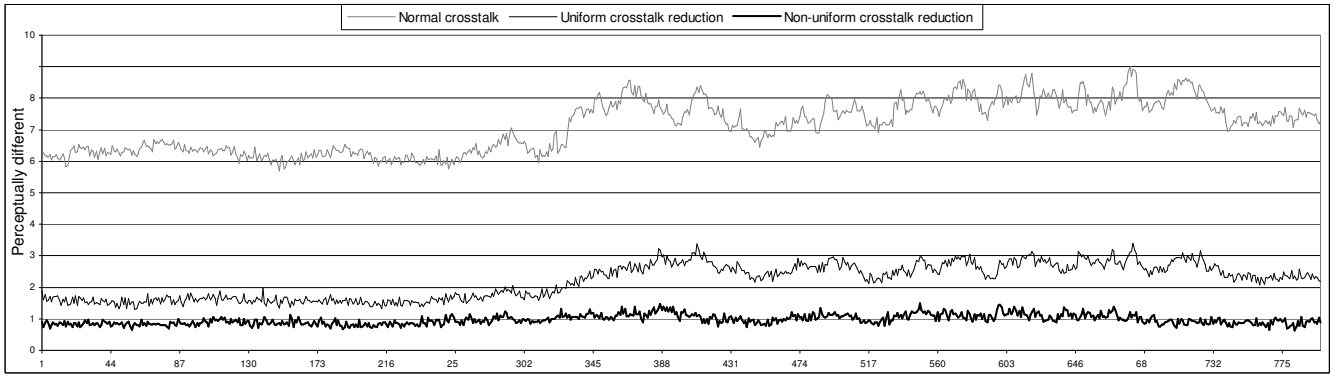


Figure 6: On the Y-axis the plot shows the percentage of perceptually different pixels compared to the corresponding reference photo. The animation frame number is shown on the X-axis.

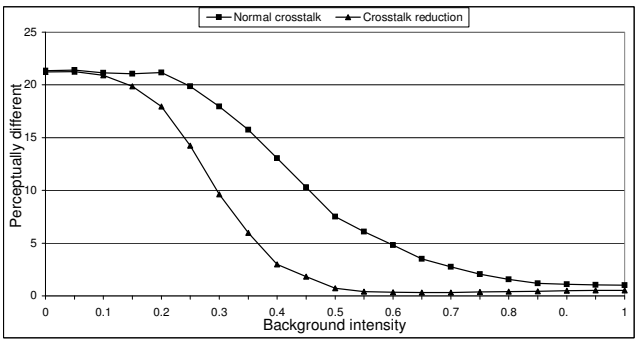


Figure 7: Percentage of perceptually disturbing crosstalk with varying background intensity, with and without crosstalk reduction.

Therefore, a second experiment consisted of determining the influence of the background intensity on the ability to perform crosstalk reduction. This time we took a sequence of photographs for frame 570 with varying background intensity. For each background intensity we took a reference photo, a photo with normal crosstalk, and with non-uniform crosstalk reduction enabled. The perceptual difference evaluation was done as before and is shown in Figure 7.

For very low background intensities, almost no correction is possible and crosstalk reduction does not provide a much better result. However, when the background intensity reaches 0.5, almost all object-to-background crosstalk can be corrected. It can also be seen that higher background intensities reduce crosstalk in general. This is due to the fact that the human visual system is less sensitive to intensity differences when intensities are large, and the phosphors reach a peak intensity value.

Finally, we experimentally evaluated the effect of different background/foreground ratios, which is defined as the number of background pixels divided by the number of foreground, or object pix-

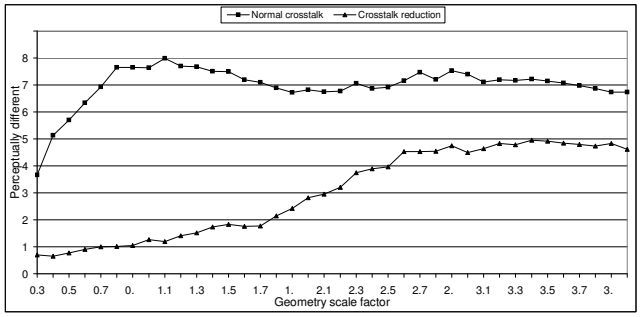


Figure 8: Percentage of perceptually disturbing crosstalk with varying background/foreground ratio, with and without crosstalk reduction. The x-axis shows an increasing geometry scale factor, which causes a decreasing background/foreground ratio.

els. Using a constant background intensity of 0.6, we varied the size of the geometry in frame 570. For larger geometry sizes the background/foregrond ratio becomes smaller, resulting in more object-to-object and less object-to-background crosstalk. Again, reference, crosstalk, and non-uniform crosstalk reduction photos were taken. The results are shown in Figure 8.

When the background/foreground ratio becomes smaller, more cases of impossible to correct object-to-object crosstalk occur. This explains why the non-uniform reduction algorithm performs worse with increasing geometry size. However, as it performs the best possible approximate correction, it still produces better quality than no correction. In the case where no correction is performed, the background/foreground ratio has less effect. Increasing the geometry size simply has the effect of replacing object-to-background crosstalk with object-to-object crosstalk, both of which are not corrected anyway. Finally, note that when the geometry becomes very small it becomes more difficult to perceive the crosstalk regions, as can be seen for the non-corrected curve. The reduction algorithm is not affected by this behavior, as it is capable of correcting the crosstalk regardless.

## 5 Discussion

Our crosstalk reduction method works well in many cases. However, in the case of heavy object-to-object crosstalk the quality of reduction is suboptimal. This is due to the fact that the amount of crosstalk reduction possible depends on the desired display intensity, i.e. when for any color channel $I_{unintended} \gg I_{desired}$, no adequate reduction can be performed. For object-to-object crosstalk this is often the case.

Unfortunately we could not quantitatively examine the effect of the temporal reduction model, but one can imagine the effect. Without the temporal model there are left video frames immediately after a buffer swap for which the crosstalk reduction is incorrect. In fact, two types of errors are introduced:

- The left video frames are not corrected for the preceding right application frames. This leaves some amount of uncorrected crosstalk, resulting in areas that are too bright.
- Instead, these left video frames are corrected for the upcoming right application frame. This crosstalk reduction on areas where it is not required introduces incorrectly darkened areas.

Manual inspection showed that this is particulary relevant for moving objects, where the edges are slightly different from frame to frame. These are exactly the regions where static crosstalk reduction methods error in their assumption of static scenes. The effect is not noticeable every frame, as application frames are drawn at a slower rate than video frames. Thus, the effect is noticeable only when application frames are changed. This typically happens at a rate of approximately 15-20 Hz and would explain the visual jitter, which is completely removed with the temporal model. We did not run a user study on this effect, but several people using our setup confirmed this observation.

Monitor-based passive stereo systems make use of a frame synchronized polarizing screen in front of the CRT display and glasses containing different polarization filters for each eye. The nature of crosstalk is very similar to active stereo in this case, however the layered construction of the polarizing screen introduces a view dependent amount of non-uniform crosstalk. This kind of crosstalk can only be compensated for by using accurate headtracking to determine the amount of correction required.

Currently the calibration of the non-uniform model is done interactively by user inspection. The procedure is tedious and error prone. However, in combination with our quantitative evaluation method it would be possible to automate this task by using a computer controlled camera. Automatic calibration would be especially desirable in virtual environments with large screens. It is a known fact that the amount of crosstalk is also dependent on the angle of view through the LCS glasses [9]. An automated calibration procedure could calibrate for many different angles of view, making use of headtracking information.

## 6 Conclusions

We have proposed a new non-uniform crosstalk model. To determine the parameters for this model, we used an interactive calibration procedure. Also, to address the problems caused by interactive applications in virtual environments, we introduced a temporal crosstalk reduction model for dynamic scenes. This way, high-frequency jitter caused by the erroneous assumption of static scenes could be eliminated. The algorithm was implemented on the GPU and runs in real-time. Finally, we proposed a quantitative evaluation methodology to assess the quality of different crosstalk reduction algorithms.

We compared our non-uniform reduction method to no crosstalk reduction, and to a uniform crosstalk reduction method. This was done by finding perceptual differences between photographs of an animation. It was shown that our method provides better quality crosstalk reduction. Finally, the effects of background intensity and background/foreground ratio were examined.

## References

[1] S. Daly. The visible differences predictor: an algorithm for the assessment of image fidelity, 1993.

[2] Electronic Industries Alliance Tube Electron Panel Advisory Council (EIA-TEPAC). Optical characteristics of cathode-ray tube screens, TEP116-C, 1993.

[3] S. Klimenko, P. Frolov, L. Nikitina, and I. Nikitin. Crosstalk reduction in passive stereo-projection systems. *Eurographics*, 2003.

[4] J. Konrad, B. Lacotte, and E. Dubois. Cancellation of image crosstalk in time-sequential displays of stereoscopic video. In *IEEE Transactions on Image Processing, Vol. 9 No. 5*, pages 897–908, 2000.

[5] J. S. Lipscomb and W. L. Wooten. Reducing crosstalk between stereoscopic views. In *Proc. SPIE Vol. 2177*, pages 92–96, 1994.

[6] A. McNamara. Star: Visual perception in realistic image synthesis. In *Eurographics 2000*, August 2000.

[7] J. D. Mulder and R. van Liere. The Personal Space Station: Bringing interaction within reach. In *VRIC*, pages 73–81, 2002.

[8] R. van Liere, J. Mulder, J. Frank, and J. de Swart. Virtual fekete point configurations: A case study in perturbing complex systems. In *VR '00: Proceedings of the IEEE VR 2000*, page 189, 2000.

[9] A. J. Woods and S. S. Tan. Characteristic sources of ghosting in time-sequential stereoscopic video displays. In *Proc. SPIE Vol. 4660*, pages 66–77, 2002.