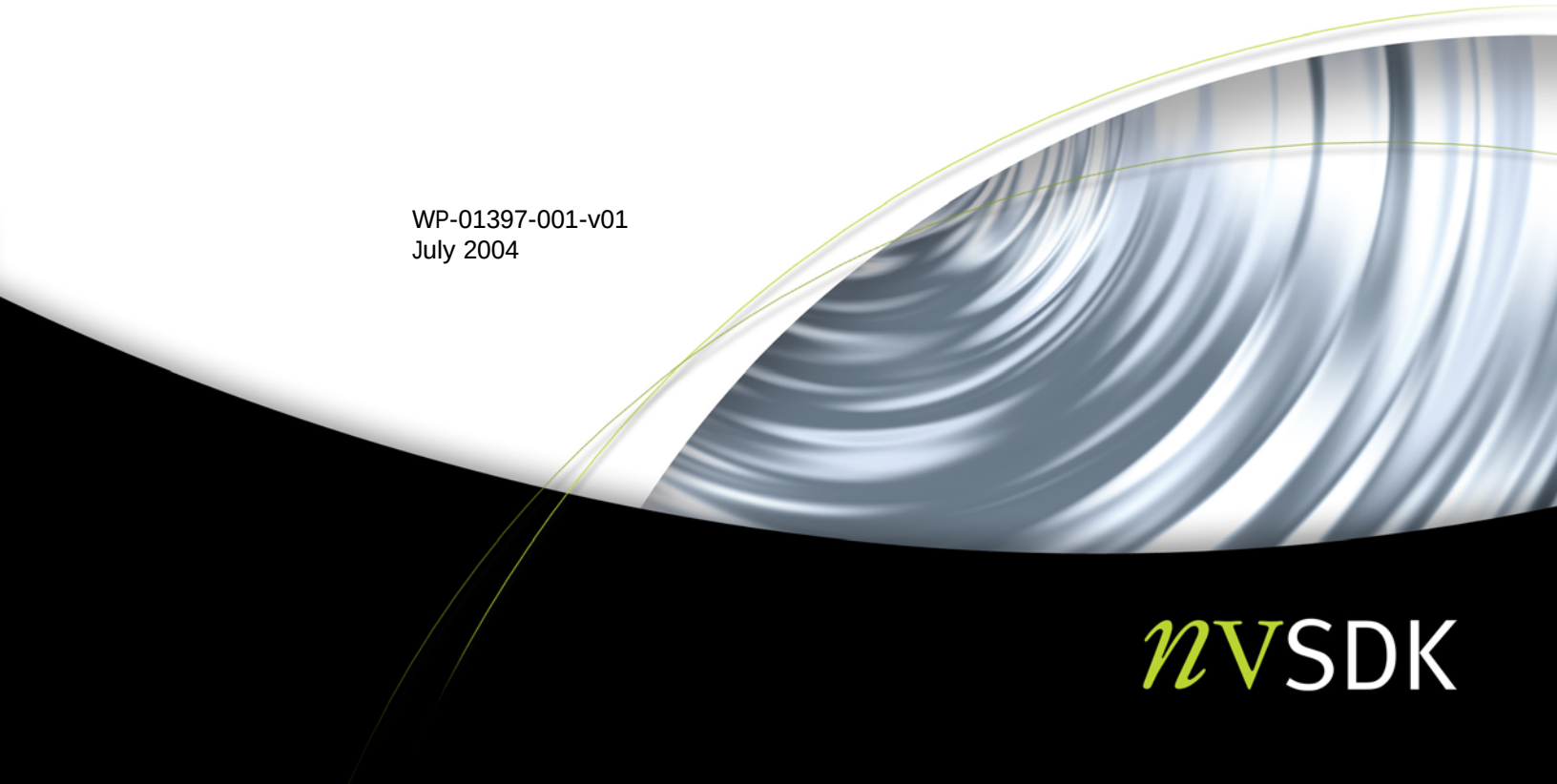
SDK White Paper

Filter Blit

WP-01397-001-v01
July 2004



Abstract

Filter Blit Example

This example demonstrates how to implement custom image processing with filters using pixel shaders to process the image. This technique can be used for a wide range of interesting filtering effects, such as blur, sharpen, and luminance edge detection.

This discussion assumes basic knowledge of graphics, and shader programming.

This example is implemented using DirectX9.0b and will run on all PS/VS1.1 class hardware.

Bryan Dudash
bdudash@nvidia.com

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050

Introduction

You can accomplish a wide array of interesting filtering effects using a filter kernel. A filter kernel is a matrix of weights for samples around the source pixel. A number of extra samples are made around each source pixel, and the results of the samples are combined using the sample weights. This is repeated for each source pixel.

The source UV is varied when picking samples. For example, consider the filter kernel in Figure 1. It samples four values on diagonals from the source pixel and averages them together evenly. This results in a blurred version of the source image.

$$\begin{bmatrix} \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix}$$

Figure 1. Simple Blur Box Filter

There is another piece of information to consider when implementing this filter kernel. The hardware does bilinear filtering when reading each sample. Thus by varying the distance offset in each direction for each sample, you can get different effective kernels. Figure 2 shows the actual and effective samples when using a texel distance of **0.5**. The reason for this becomes clear when you look at the equations for each sample. With each diagonal equally averaging the four texels it touches, and then each sample being averaged equally, you obtain the resultant effective **9** sample cone filter in Figure 2.

$$S_0 = (a+b+d+e)/4 \qquad S_2 = (d+e+g+h)/4$$
$$S_1 = (b+c+e+f)/4 \qquad S_3 = (e+f+h+i)/4$$

$$P = (S_0 + S_1 + S_2 + S_3)/4$$
$$\therefore P = e/4 + (b+d+f+h)/8 + (a+c+g+i)/16$$

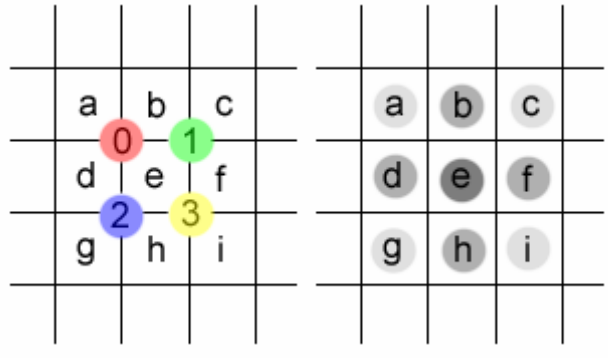


Figure 2. Simple Box Filter and Effective Filter Kernel Samples

In Figure 2, the diagram on the left shows the actual samples as defined by the simple box filter. The diagram on the right shows the effective filter kernel obtained when using a **0.5** texel offset on the samples.

Other Blur Filters

A general equation for the actual filtering equation using the simple box filter with a varying sample distance is:

$$P = \frac{s^2}{4}(a + c + g + i) + \frac{(s - s^2)}{2}(b + d + f + h) + (1 - s)^2 e$$

By changing the sample offset distance to **2/3**, you effectively change the weight of all **9** samples, and get the box filter shown in Figure 3.

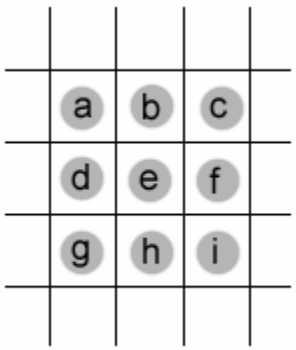


Figure 3. Nine Sample Box Filter Obtained with a Texel Offset of 2/3

A **16** sample filter can be obtained by extending the texel distance even further and placing each sample on the corner of **4** independent samples. In that case, the bi-linear hardware will evenly filter **4:1** for each sample, and the pixel shader can combine those samples.

Luminance Edge Detection Basics

The Luminance Edge Detect algorithm as implemented to show 3 different luminance implementations as indicated in Figure 4 below.

1. Color to Luminance (**`Luminance`**)
2. Luminance multi-sample edge calculation (**`LuminanceSensitiveDiagEdge`**)
3. Luminance multi-sample edge calculation and blending (**`LuminanceDiagEdge`**)

All the above implementations use a luminance conversion vector to convert a color value to a luminance value. The edge detect calculation samples the diagonals off the center, and then calculates slopes on those diagonals. It multiplies the result by a large number to make the values visible, and outputs the result. Please refer to the shader for more specifics.

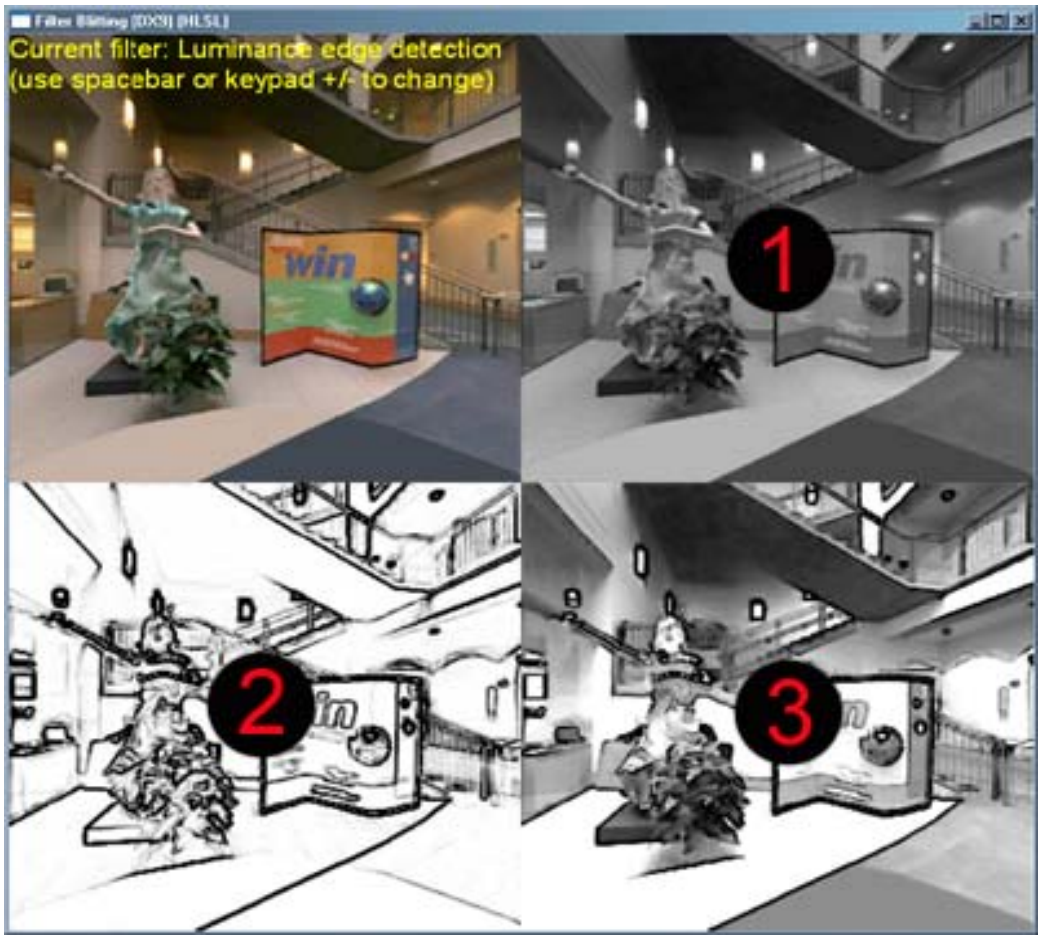


Figure 4. Luminance Edge Detection implementations

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com