A Steerable Streak Filter

Christopher Oat
3D Application Research Group
ATI Research

Introduction

This article presents an image filtering technique for generating directional streaks across an image from light sources or bright reflections. This technique is based upon work done by Masaki Kawase for the XBox game *Double-S.T.E.A.L.* (called *Wreckess: The Yakuza Missions* outside of Japan) and described at CEDEC 2002 [Kawase02]. What makes our extension of this technique interesting is our use of a filter kernel which is *steerable* [Freeman91]. That is, the filter kernel can take on different shapes and orientations at different locations in the image in order to simulate diffraction or to further stylize the image.

Background

Anyone who has driven at night is probably familiar with how the light emitted from the headlights of on coming traffic streaks across one's field of view, seeming to smear across the windshield itself. Imperfections in the car's windshield, frequently the result of windshield wipers scratching small grooves in the glass, scatters incoming light in a way that causes it to appear to flare out from its source [Nakamae90].

Light streaks are the result of light scattering as it passes through a lens or aperture. Cracks, scratches, grooves, dust and lens imperfections cause light to scatter and diffract. Light has many opportunities to scatter as it travels from its source to the photoreceptors in the viewer's eye or a camera. Our eyelashes and even the eye's lens can diffract and scatter incoming light. The more intense the incoming light is relative to the surrounding ambient light level, the more perceivable the scattering will be. Because of this, light streaks are an important visual cue that help the viewer discern the relative brightness of light sources in comparison to their surroundings [Spencer95].

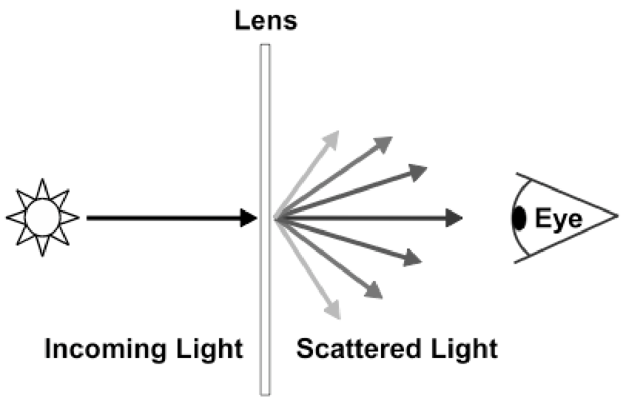

Figure 1. Light scatters as it passes through glass. Surface imperfections in the glass determine in which direction light rays will scatter.

As shown in Figure 1, the amount of scattering and the orientation of the scattered light rays depend on the orientation and physical properties of the media through which the light is being transferred. Rather than simulate the complex light transport that results in light streaks (which would require computationally expensive ray tracing algorithms), we seek to mimic the perceivable end result using a post-process image filter. Our method for replicating the perceivable effects of light streaking is an image space technique that relies on a steerable streak filter. This filter is intended to be applied to a rendered scene image and results in directional streaking of bright lights and reflections across the image. The results of the filter are composited back onto a rendered image before presenting the final result to the user.

Steerable Streak Filter

Like all image-space techniques, the streak filter uses multiple rendering passes that each draw a full screen quad to an off-screen render target using various renderable textures as input. Figure 2 illustrates the high level algorithm for applying the streak filter to a rendered scene.

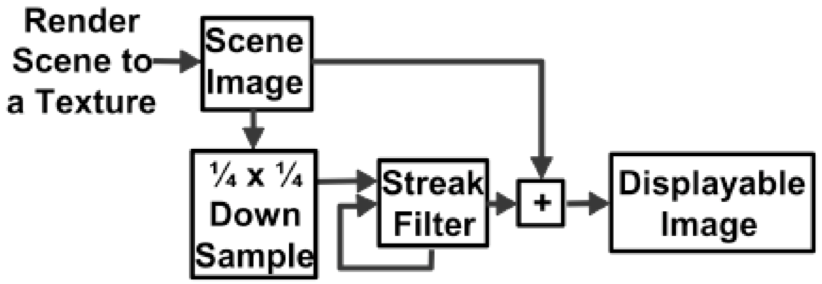

Figure 2. Applying the streak filter to a rendered scene: Render scene to a texture, down sample, iterate the streak filter several times, composite streaks back onto scene image and then finally display the composite image.

We begin by rendering the scene to an off-screen render target that we call the "scene image." Next, the scene image is downsampled using a box filter. This downsampling step reduces the number of pixels that will subsequently be processed by the streak filter, thus increase performance. As a side effect, the streaks will be slightly more blurred, which is generally desirable. The first pass of the steerable streak filter takes the downsampled scene image as input and renders into an additional off-screen buffer. The streak filter works by "ping-ponging" between two off-screen buffers. That is, one buffer is used as input while the other is used as output. After each iteration of the filter, the roles of these off-screen ping-pong buffers switch (what was previously the input buffer becomes the new output buffer, etc). Thus, the results of one iteration of the filter may be used as the input for the next iteration of the filter. Each pass may be thought of as a progressive refinement that increases the smoothness and length of the streaks. The filter kernel shape and sample weights change slightly for each iteration.

First Pass

As shown in Figure 3, the kernel operates on four samples from the downsampled scene image.

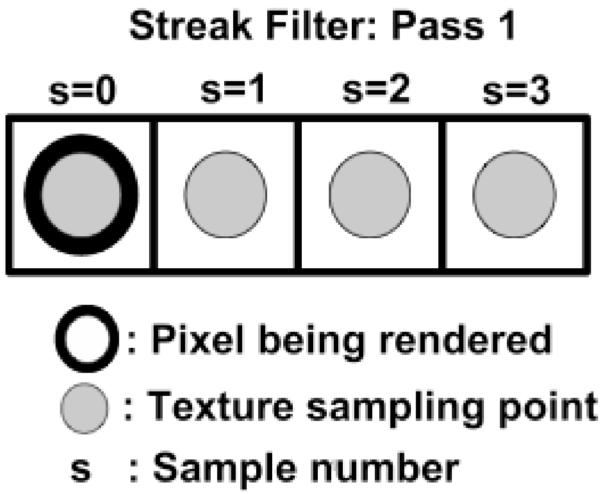

Figure 3. The streak filter takes four samples, one sample at the location of the pixel being rendered and three additional samples in the direction of the streak.

The first sample is taken at the screen location of the pixel being rendered. Three additional samples are taken from neighboring pixels. The neighboring pixel locations are determined according to the orientation of the streak. Each neighboring sample is weighted and attenuated according to its sample number, a constant attenuation and the pass number.

$$weight = a^{(b*s)}$$

$$a = attenuation = [0.9, 0.95]$$

$$b = 4^{(pass-1)}$$

$$s = sample\_number$$

The attenuation is a constant term that may be tweaked to adjust the strength of the streaks. Generally, an attenuation term in the range 0.9 to 0.95 gives reasonable results. The term *b* above is the screen space distance between sample points, for each pass of the filter the sample points are taken further from the pixel being rendered. Figure 4 illustrates the progression of the streak filter for multiple passes.

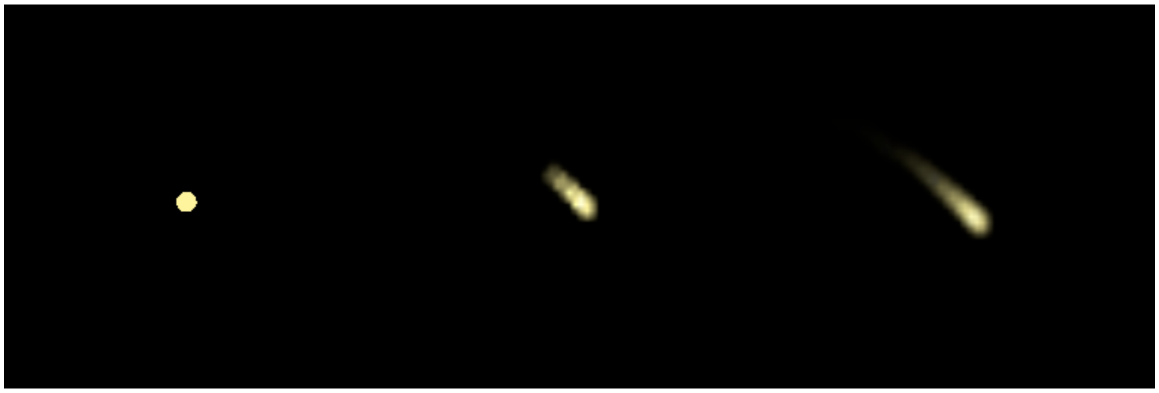
Figure 4. Progression of the steerable streak filter. (Left) The scene image. (Middle) Intermediate results after a single iteration of the filter. (Right) Results from two iterations of the streak filter.

Second Pass

For each pass of the filter, the kernel is expanded in the direction of the streak. Figure 5 illustrates the shape of the streak filter's kernel during the second pass of the filter.

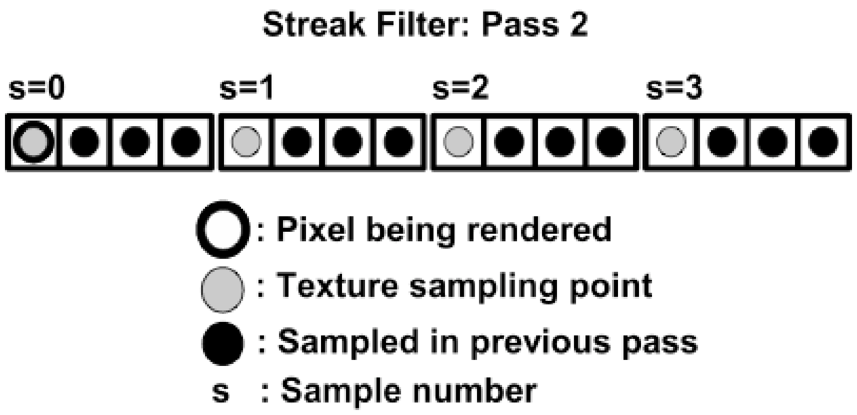

Figure 5. The second pass of the streak filter expands the kernel in the direction of the streak. Four samples are taken, one at the location of the pixel being rendered and three additional samples at every fourth neighboring pixel in the direction of the streak.

Each successive pass of the filter further expands the kernel in the direction of the streak. The kernel expands by sampling increasingly distant neighbors. For a given pass, the distance between neighbors is specified by the term *b*. As you can see, the overall size of the kernel grows for each pass. For a given pass, the distance between neighbor samples is equal to the size of the kernel from the previous pass (the squared distance from the previous pass). While each pass uses an increasingly larger kernel than the previous pass, the individual sample weights decrease as their distance from the pixel being rendered increases. Frequently, only two passes are necessary to achieve acceptable results. Of course, this is dependant on the resolution of the source image and the size of the streaks you wish to generate.

So far, our streak filter only streaks in a single direction as specified by the 2D per-pixel streak orientation vector (sampled from the streak orientation texture). Applying this process in only a single direction results in streaks with only a single "radial arm." For a more interesting effect with several streak directions, the multi-pass streak filter must be executed several times (one execution for each per-pixel streak direction). When applying the streak filter multiple times for multiple streak directions, it is important to keep a clean version of the down-sampled scene image around. If you reuse the downsampled scene image as one of the ping-pong render targets, then you won't have the original downsampled scene image to use for a the next application of the streak filter. Since each application of the streak filter requires multiple passes, the number of streak filter applications should be kept small. For most scenes, four applications of the streak filter should produce reasonable results (4 streaks × 2 passes per streak = 8 passes total).

Streak Direction

As mentioned earlier, the important property of our extension to Kawase's streak filter is steerability. Our filter kernel may be oriented on a per-pixel basis, allowing us to vary the direction in which the streaks will radiate. Streak orientation may be specified on a per-pixel basis by sampling an orientation texture. The orientation texture may correspond to the surface properties of a scratched windshield or some other scattering media. Orientation vectors are packed into a texture by placing vector components into the color channels of each texel. The streak orientation texture will be screen aligned and sampled during each pass of the streak filter. Because we are working in image space, only 2D streak directions are needed. Thus, only two channels of the streak orientation texture are needed to store orientations. The remaining channels may be used to store such things as transparency masks or per-pixel streak attenuation values. Figure 6 illustrates two possible streak orientation textures.

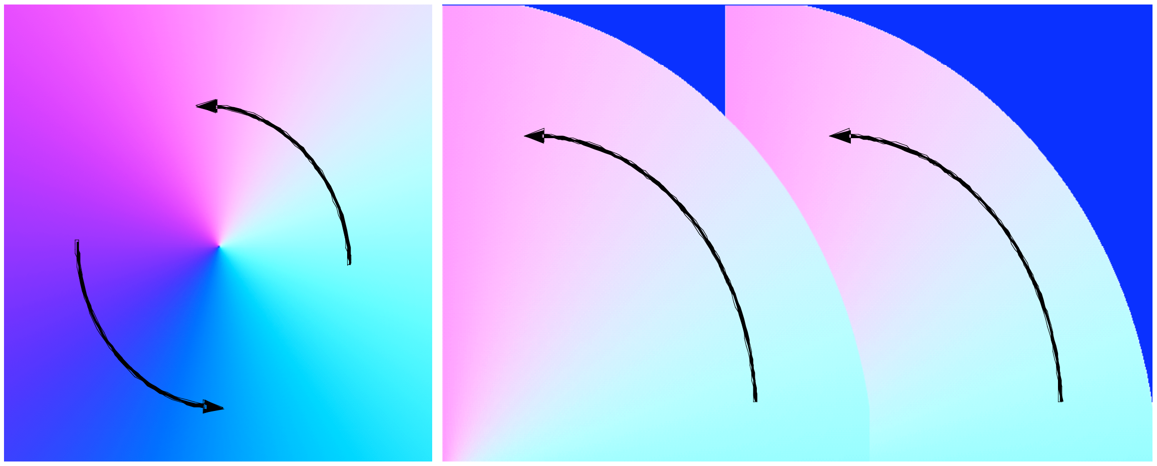
Figure 6. 2D streak orientations are stored in the red and green channels of a texture. (Left) A circular streak orientation texture. (Right) A streak orientation texture that mimics scratches that windshield wipers leave on a car's windshield.

To save texture memory, it may be desirable to only store a single streak orientation texture even if you wish to apply the streak filter multiple times for multiple streaks. A single streak orientation texture may be reused for each successive application of the streak filter by rotating the streak orientation vector by some fixed amount for each application of the streak filter. For example, if four streaks per-pixel are desired, the streak filter algorithm must be applied four times, each time the per-pixel streak direction should be rotated by 90 degrees. Streak directions are 2D vectors in screen space so rotating them is simply a matter of rotating about the *z* axis (the axis perpendicular to the screen). Each application of the streak filter will produce streaks oriented in slightly different directions. These streaked images may then be composited back onto the scene image. Figure 7 demonstrates four applications of the streak filter, rotating the per-pixel streak direction by 90 degrees each time and blending the results of each application.

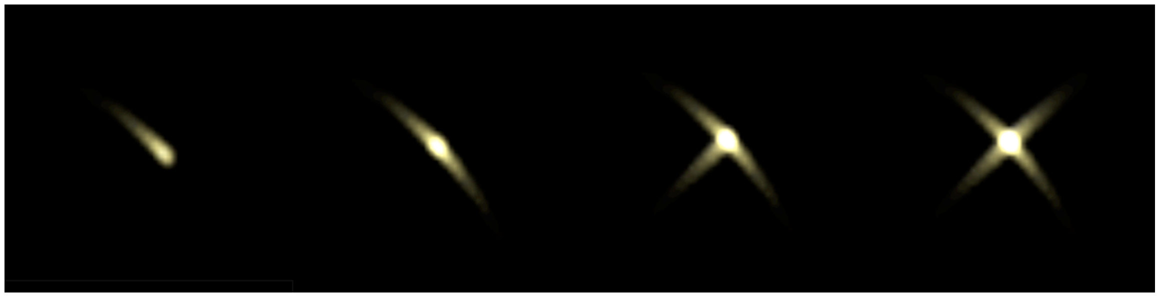
Figure 7. The streak filter is applied four times to the same scene image (progressing from left to right). Each application of the streak filter rotates the streak direction by 90 degrees and the results of each application of the filter are blended to create a final composite with four streak directions.

Pixel Shader

Implementing the streak filter is simply a matter of writing some pixel shader code to compute the sample points and weights. The streak filter is implemented here as an HLSL function and may be called from a pixel shader. It is assumed that the caller of this function has sampled the streak orientation texture and will determine the attenuation value.

```
#define NUM_SAMPLES 4
float3 SteerableStreakFilter (sampler tSource, float2 texCoord,
                              float2 pxSize, float2 dir,
                              float attenuation, int pass)
{
   float2 sampleCoord = 0;
   float3 cOut = 0;

   // sample weight = a^(b*s)
   // a = attenuation
   // b = 4^(pass-1)
   // s = sample number

   float b = pow(NUM_SAMPLES, pass);

   for (int s = 0; s < NUM_SAMPLES; s++)
   {
      float weight = pow(attenuation, b * s);

      // dir = per-pixel, 2D orientation vector
      sampleCoord = texCoord + (dir * b * float2(s,s) * pxSize);
      cOut += saturate(weight) * tex2D (tSource, sampleCoord);
   }

   return saturate(cOut);
}
```

Results

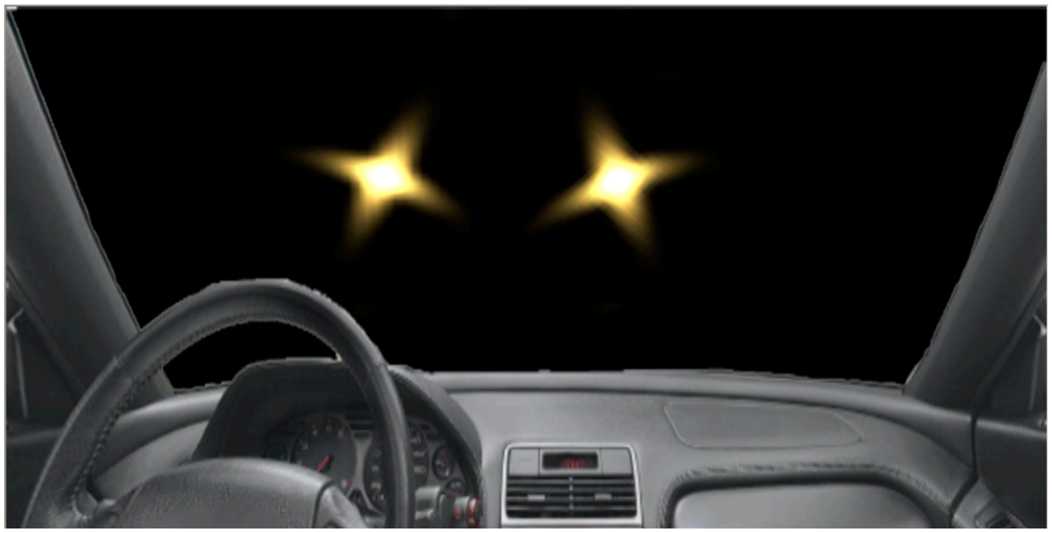

Figure 8. The steerable streak filter was used to streak the headlights of oncoming traffic in this image.

The steerable streak filter was used to produce the image shown in Figure 8. Headlight geometry was drawn into a scene image and then downsampled using a standard box filter. A two-pass streak filter was applied four times to produce four streaks radiating from the headlights. The streak orientation was specified using the streak orientation texture (from Figure 6) that mimics the scratches in a car's windshield. As you can see, the four radial arms of the streaks caused by the two headlights are oriented differently. Some of the radial arms are even bending according to the grooves in the car's windshield as encoded in the streak orientation texture.

Conclusion

This article presented a steerable streak filter that can be used to simulate the diffraction of light caused by windshields, lens apertures and even our eyelashes. Light streaks are a popular stylistic element and can be an important visual cue for perceiving the relative brightness of a light source such as the approaching headlights of oncoming traffic as seen through a car's windshield. A major advantage of this filter is that it can be oriented on a per-pixel basis via a streak orientation texture.

References

[Freeman91] William T. Freeman and Edward H. Adelson, "The Design and Use of Steerable Filters," IEEE Trans. Pattern Analysis and Machine Intelligence, 13:891--906, 1991.

[Kawase02] Masaki Kawase. "Double-S.T.E.A.L Techniques," CEDEC 2002. http://www.daionet.gr.jp/~masa/column/2002-09-22.html

[Nakamae90] Eihachiro Nakamae, Kasufumi Kaneda, Takashi Ocamotoand Tomoyuki Nishita, "A Lighting Model Aiming at Drive Simulators," SIGGRAPH 1990, pp. 395-404.

[Spencer95] Greg Spencer, Peter Shirley, Kurt Zimmerman, Donald P. Greenberg, "Physically-Based Glare Effects for Digital Images," SIGGRAPH 1995, pp. 325-334.