

Visualizing the 4D Blur Matrix

Kevin Simmons
Computer Science
Alabama A&M University

Faculty Mentor: Brian A. Barsky
Graduate Mentor: Todd J. Kosloff

Summer Undergraduate Program in Engineering Research at Berkeley –
Information Technology (SUPERB-IT) 2007

Department of Electrical Engineering and Computer Sciences
College of Engineering
University of California, Berkeley

Visualizing the 4D Blur Matrix

Kevin Simmons
Computer Science
Alabama A&M University
kbs1785@hotmail.com

Abstract

Computer graphics can be classified into subfields of geometry, animation, rendering, and imaging. Rendering has uses in architecture, video games, TV special effects, and design visualization, each employing a different balance of features and techniques. Many rendering algorithms have been researched, and software used for rendering may employ a number of different techniques to obtain a final image. The purpose of my project is to create images of the blur matrix. The user will see shades of grey which means the image will transition from white to gray and then finally to black. My program will have a graphical user interface that will include buttons, sliders, and checkboxes that will play a key role in displaying the image. I will be using various mathematical equations and algorithms that will be implemented into the C++ programming language.

Introduction

Cameras and eyes typically don't image a scene entirely in perfect focus. The region within which objects appear sharp, or "depth of field" is limited. The depth of field effect has been simulated in computer graphics, but with the same limited control as found in real camera lenses [2]. Our focus is on developing synthesized images. We want these images because they provide the ability to direct the viewers' attention to a particular segment of an image. Another reason is because the images permit adaptation of many well known used "cinematographic techniques for animated sequences, such as fade in, fade out, uniform defocusing of a scene, depth of field, lens distortion, and filtering [6]." The process of blurring acts as a low-pass filter by removing high area frequencies which cause surprisingly unexpected changes.

We want to make an image that shows a 4D description of our blur. We introduce the concept of using a 4D matrix to perform depth of field post processing. Post processing methods use image processing to add blur to a pinhole render. Depth of field was already thought of before we began our project. Depth of field is the swath through a 3D scene that is imaged in acceptable focus through an optics system, such as a camera lens. It is the distance in front of and beyond the subject that appears to be in focus. This paper explores the techniques of creating different images.

Background Information

Some fast blur filters do not work well for depth of field. Fast Fourier transform (FFT), which is an efficient algorithm that computes the discrete Fourier transform (DFT) and its inverse, is less useful when the amount of blur varies throughout an image. Some fast filters operate by gathering. Gathering calculates the color of an output pixel by summing over the region of the input image. The region to sum over ranges over a large area or small area depending on whether that output pixel should be more or less blurred. The desired blurriness of the neighboring pixels is ignored due to the fact that gathering bases the size of the region to sum over only on the output pixel. A very blurred pixel will not correctly receive contributions from neighboring in-focus pixels or vice versa. Gathering can not be performed if a pixel is outside an object.

Some filters use the technique of spreading. Spreading involves each input pixel spreading out into a point spread function. Spreading usually results in blurred objects that have soft edges. A single output pixel can receive contributions from an enormous amount of input pixels and each of these input pixels might have a totally different-sized PSF.

In order to get accurate results we decided to download a toolkit that would be beneficial to us creating an outstanding Graphical User Interface (GUI). This toolkit is known as the Fast Light Tool Kit (FLTK). Fast Light is a C++ GUI toolkit that is used to support 3D graphics. Some benefits of FLTK are that it's Free Open Source Software, which means that it is open to the public. FLTK is also an asset to us because it uses C++, it supports OpenGL, and it has a graphical user interface builder called FLUID. I thought FLTK was very useful because it offered tutorials on how to create buttons and sliders.

Equations used to get results	Formula
Distance	$(u-s)^2+(v-t)^2$
Blur Matrix	$A * e^{-((u-s)^2+(v-t)^2)}$
Sine Wave	$C*\sin(D*s)+C;$

Results

In the C++ program that I created it contains two push buttons, eight radio buttons, and two sliders. My program has six different test patterns, two modes, and four dimensions. When the user clicks a radio button in the test pattern category, a radio button in the mode category, two checkboxes in dimension, and then the push button; the program will display an image to the user. When the user views the program they probably are thinking what is the purpose of the four letters? These four letters u, s, t, and v represent the x and y coordinates of the input and output images. My program visualizes four dimensions on a 2D screen by holding two of the dimensions constant while mapping the other two to the screen. The user can only check two checkboxes at one time while the two

checkboxes are checked the other two will be disabled until there is a change in dimension. The program also includes a feature called mode that consists of an overview of an image or a detailed display of the image. The process of creating the mode feature and implementing it into the program was definitely one of the hardest tasks of creating the project. The detail section was developed through me creating a `make_image()` function and the overview image was developed from me creating a `make_image2()` function. The image in Figure 1 displays a test pattern image titled 1st wave, it is in overview mode and has the s and t dimensions controlled by the sliders which means u and v are controlled by the x and y position of the screen. Slice (s,t) use the spreading technique.

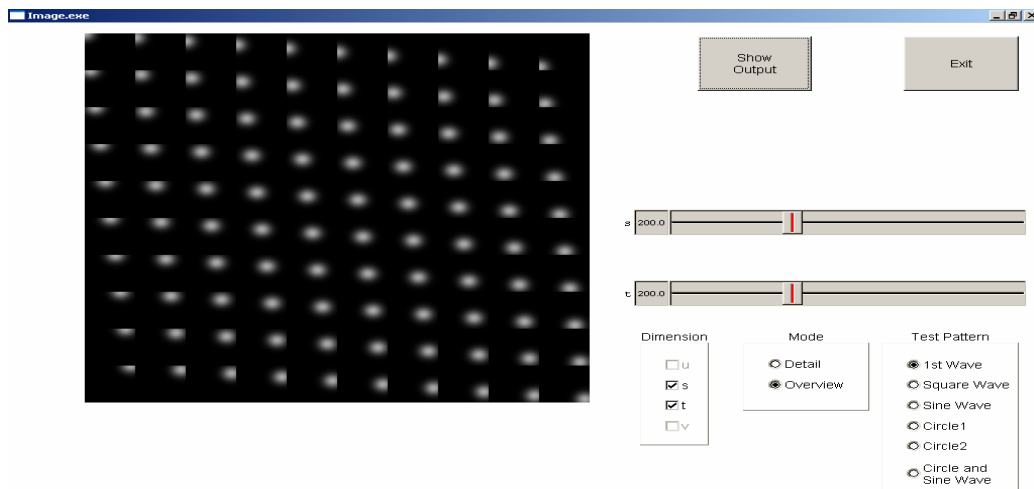


Figure 1: Test Pattern - 1st Wave. Mode - Overview. Dimension – s, t.

The image in Figure 2 displays a Square Wave Test Pattern, it is in overview mode and has the u and v dimensions controlled by the sliders which means s and t are controlled by the x and y position of the screen. Slice (u,v) displays a curvy edge from using the gathering technique.

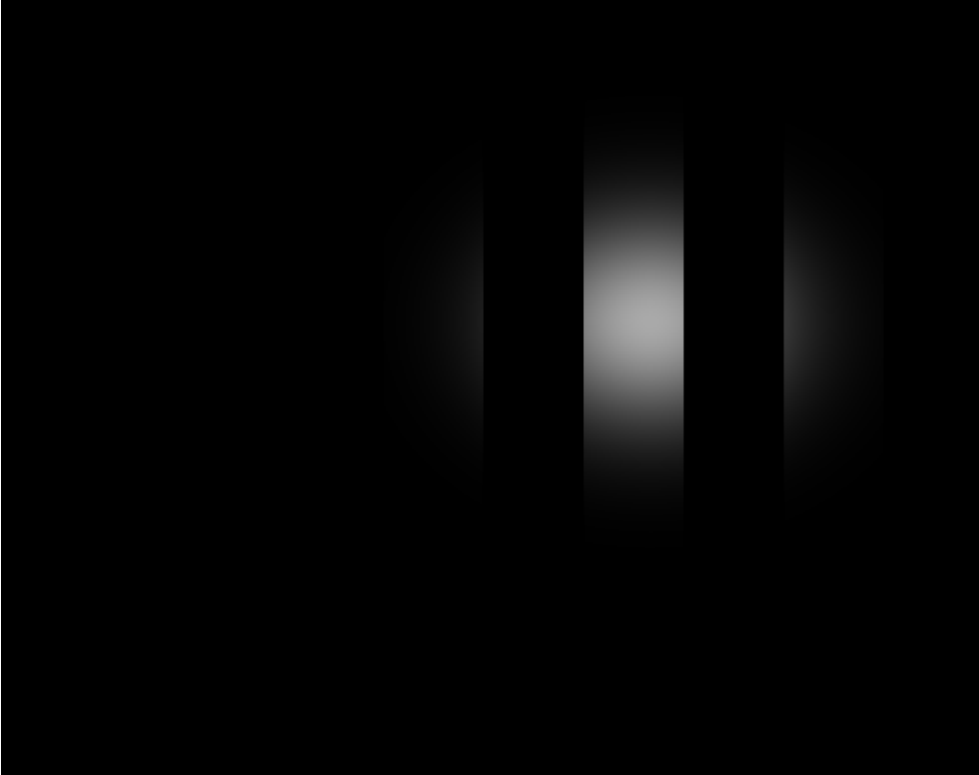


Figure 2: Test Pattern – Square Wave, Mode – Detail, Dimension – u and v

The image in Figure 3 displays a Sine Wave Test Pattern, it is in detail mode and has the u and v dimensions controlled by the sliders which means s and t are controlled by the x and y position of the screen.



Figure 3: Test Pattern – Sine Wave, Mode – Detail, Dimension – u and v

The image in Figure 4 displays a Sine Wave Test Pattern, it is in overview mode and has the s and t dimensions controlled by the sliders which means u and v are controlled by the x and y position of the screen.

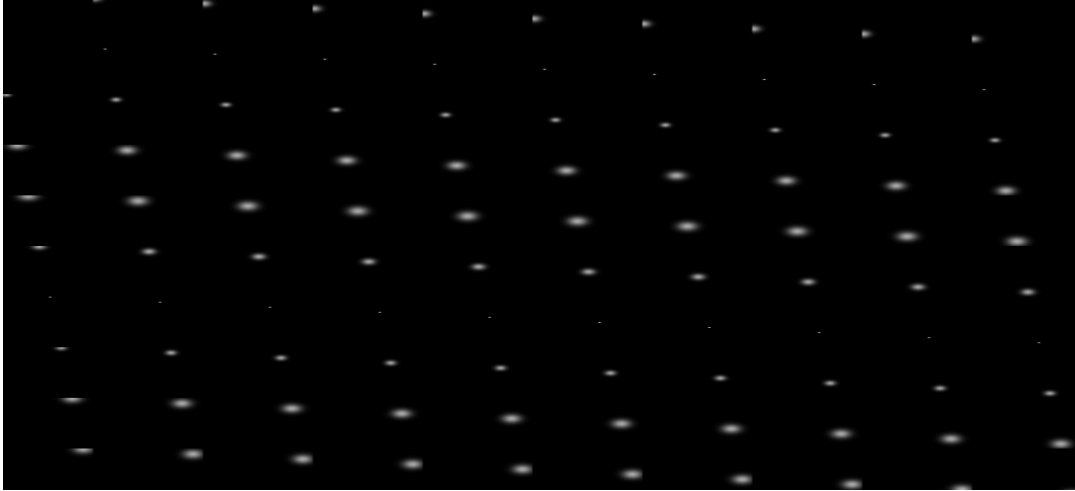


Figure 4: Test Pattern – Sine Wave, Mode – Overview, Dimension – s and t

The image in Figure 5 displays the Circle 1 Test Pattern, it is in overview mode and has u and v dimensions controlled by the sliders which means s and t are controlled by the x and y position of the screen.

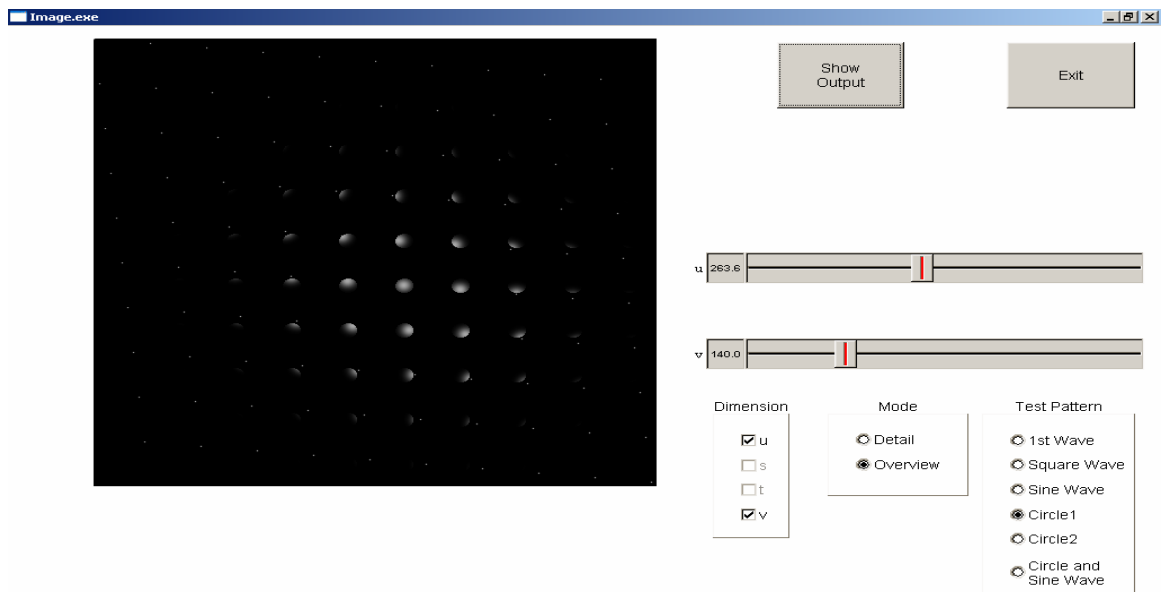


Figure 5: Test Pattern - Circle 1, Mode – Overview, Dimension – u and v
For Test Pattern Circle 2 I did the opposite. I said if the pixels are inside the circle it will be a significant amount of blur but if the pixels are outside the circle the blur will be a less amount.

The image in Figure 6 displays the Circle 2 Test Pattern, it is in overview mode and has s and v dimensions controlled by the sliders which means u and t are controlled by the x and y position of the screen.



Figure 6: Test Pattern - Circle 2 , Mode – Detail, Dimension - s and v

The image in Figure 7 displays the Circle and Sine Wave Test Pattern, it is in detail mode and has t and v dimensions controlled by the sliders which means u and s are controlled by the x and y position of the screen. Slice (t,v) is similar to (u,s)



Figure 7: Test Pattern - Circle and Sine Wave, Mode – Detail, Dimension – t and v

The image in Figure 8 the image displays the Circle and Sine Wave Test Pattern. The figure is saying that u and v are sliders while s and t are represented by x and y .

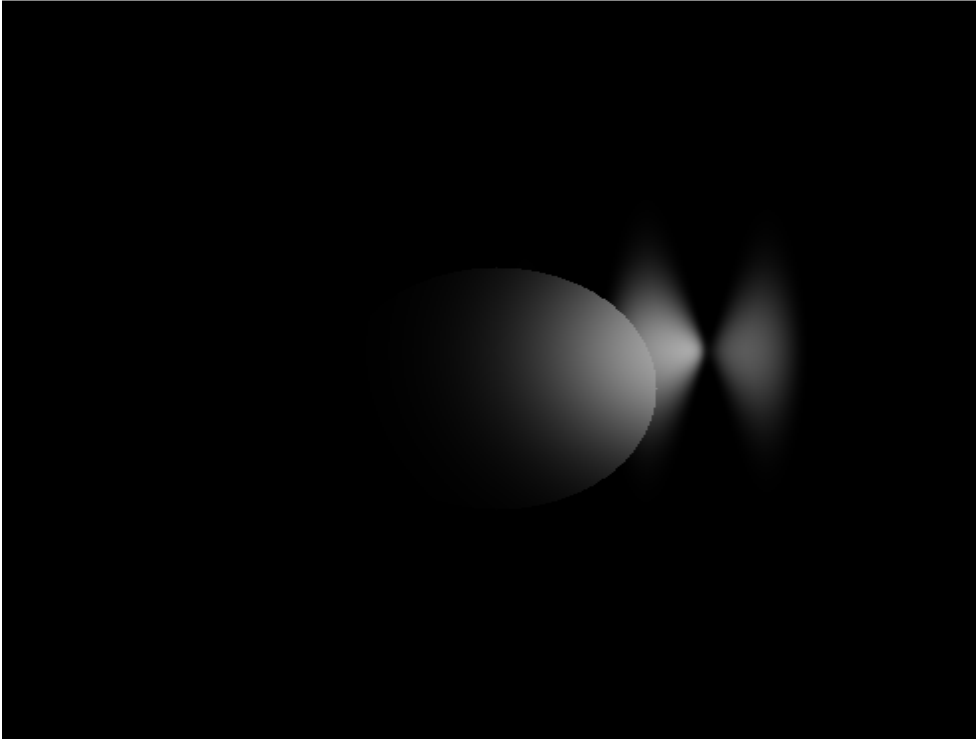


Figure 8: Test Pattern – Circle and Sine Wave. Mode - Detail. Dimension – u, v .

In Figure 9 the image displays the Sine Wave Test Pattern. The figure is saying that u and v are sliders while s and t are represented by x and y .

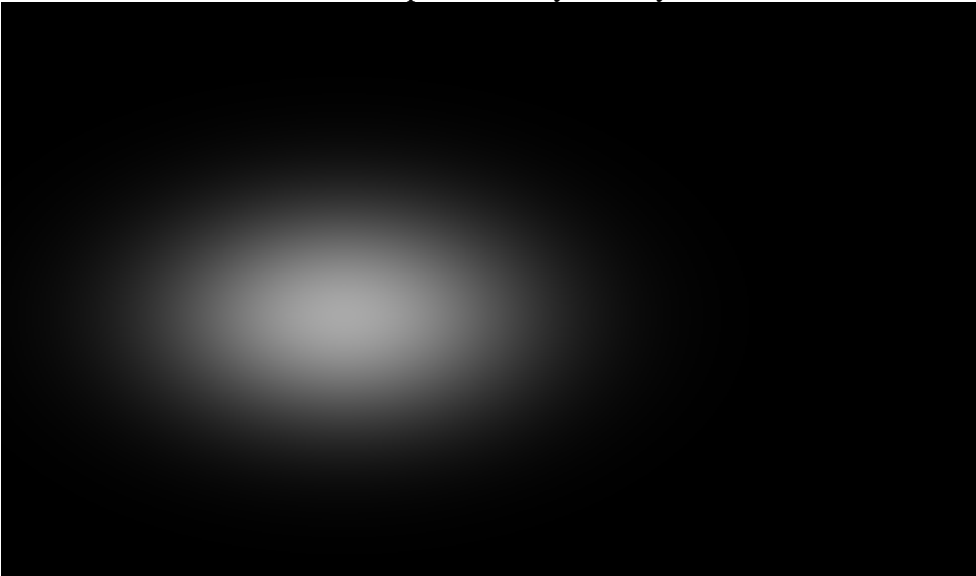


Figure 9: Test Pattern – Sine Wave. Mode - Detail. Dimension – u, v .

In Figure 10 the image displays the Square Wave Test Pattern. The figure is saying that u and s are sliders while t and v are represented by x and y . After studying slice (u,s) , we recognize that if the values of the sliders are exactly the same then the output is the brightest.

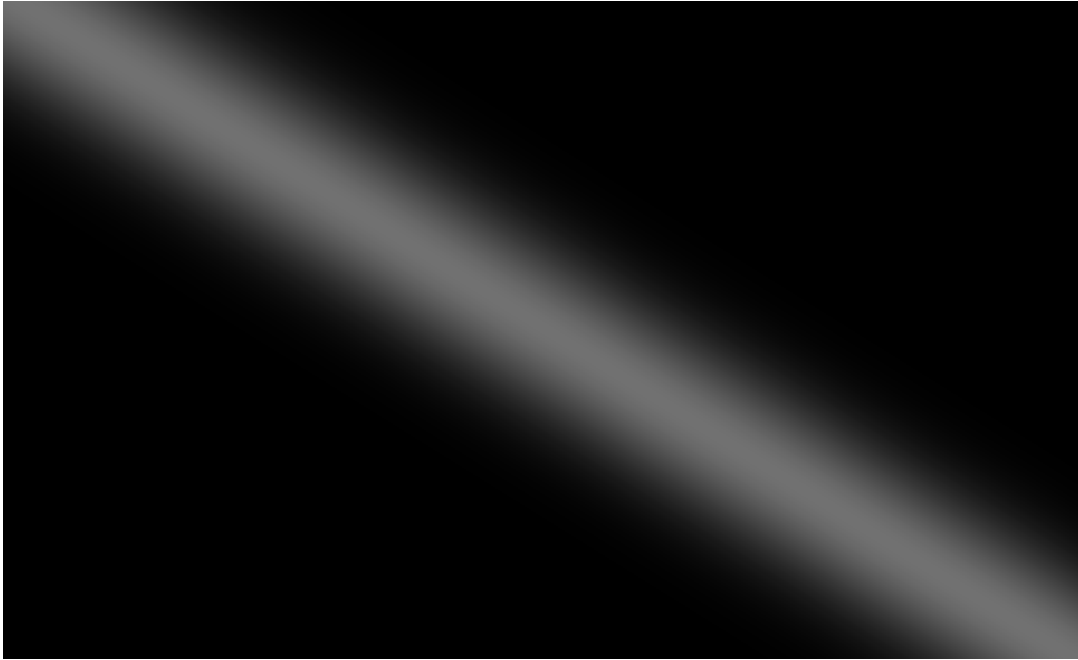


Figure 10: Test Pattern – Sine Wave. Mode - Detail. Dimension – u, s .

Conclusion

We learned that the point spread function for each point will be different because there are different test patterns. We believe that we chose the right toolkit for this project. Thanks to FLTK we have made significant progress with our results. The mathematical formulas that we implemented showed us some incredible blurred images. We were able to analyze and interpret most of the images we created. Lastly we successfully created 4D images of the blurred matrix. The combination s and t is a 2D picture linked to the combinations of u and v which is another 2D picture together they create 4D images of the blurred matrix. From doing this research we learn about the next step, which focuses on how blurring could use insight learned from the illustration to create better algorithms. The (s,t) slice might be better than (u,v) slice so we would want a program that will analyze (u,v) more.

Acknowledgements

I would like to like my faculty mentor Dr. Brian Barsky, my graduate mentor Todd Kosloff for his guidance, the SUPERB-IT program for selecting me to participate in an exciting research experience, The University of California at Berkeley, and the National Science Foundation.

References

1. Barsky, Brian. A. "Vision Realistic Rendering: "Simulation of the Scanned Foveal Image with Elimination of Artifacts due to Occlusion and Discretization"" Computer Science Division and School of Optometry University of California, Berkeley
2. Kosloff, Todd. J., Barsky, Brian. A. "An Algorithm for Rendering Generalized Depth of Field Effects Based on Simulated Heat Diffusion" 24 January 2007 < <http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-19.pdf>
3. Heckbert, Paul S. "Filtering By Repeated Integration", Pacific Data Images Sunnyvale, CA Vol. 20, No 4, Dallas Aug 18-22, 1986
4. "Fourth Dimension" 12 June 2007 Wikipedia Foundation Incorporated < http://en.wikipedia.org/wiki/Fourth_dimension
5. Haeberli, Paul. Akeley, Kurt. Silicon Graphics Computer Systems "The Accumulation Buffer: Hardware Support for High-Quality Rendering", Computer Graphics, Vol. 24, No 4. Aug 1990 < <http://delivery.acm.org/10.1145/100000/97913/p309-haeberli.pdf?key1=97913&key2=8897565811&coll=GUIDE&dl=portal,ACM&CFID=25222727&CFTOKEN=25100403>
6. Potmesil, Michael., Chakravarty, Indranil "Synthetic Image Generation with a Lens and Aperture Camera Model" Research Contributions. ACM Transactions on Graphics, Vol. 1, No 2, April 1982, Pages 85-108