Real-Time Generation of Autostereoscopic Images

Scott Grauer-Gray
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
grauerg@cis.udel.edu

Abstract

This paper explores the creation of autostereocopic images using a light field with a focus on real-time generation of these images using the GPU.

1. Introduction

The generation of autostereoscopic images can be a challenging problem in many situations. One method of generation is to generate an integral photograph where an array of packed-together lensets is used to capture the desired autostereoscopic display. However, there are complications with this method as descibed by Isaksen in [2]. Isaksen goes on to describe a novel method of autostereoscopic image generation via a reparameterization of a light field. We explore the possibility of real-time autostereoscopic image generation using this method.

2. Light Field Framework Using Reference Image Database

In [2], Isaksen presents a light field parameterization described with a camera surface and a focal surface, in which a mapping between the focal surface and the image plane for each camera is used to determine the view of the scene from a virtual camera given a database of reference images. This flexible parameterization is shown in figures 1 and allows for effects such as variable aperture, variable focus, and multiple apertures and focal surfaces.

3. Autostereoscopic Image Generation Using Light Field

Isaksen goes on to show how this camera surface-focal surface parameterization can be used to generate a direct viewing of the light field, which is essentially

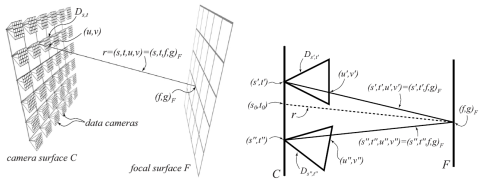

Figure 1: Display of light field parameterization presented by Isaksen.

equivalent to the creation of an autostereoscopic display of the image. The view from each camera is generated for each of a set of points on the focal surface. The set of views from each camera for a given focal surface point are combined to form a sub-image in the autostereoscopic display, and the sub-images representing each view are combined to form the output autostereoscopic image.

3.1. Real-Time Generation of Autostereoscopic Image

We explore the generation of autostereoscopic images on the CPU and the GPU, looking at the possibility of real-time generation of autostereoscopic images at dynamically changing focal surfaces and rendering a variable number of points and locations on the focal surface. In our implementation, we render from every camera in the given database of images. In cases where the view of a focal surface point from given camera is not available, the view is set to a default 'black' color for that camera.

3.2. Varying the number and location of focal surface points

Our implementation allows for the user to easily adjust the number and the outer bounds of the rendered

focal surface points. The choice of more points gives more information about the image, but can result in a larger output image than desired and takes longer to process, while rendering using fewer surface points allows for quicker generation of autostereoscopic images at the expense of less information about the scene.

3.3. Varying the focal surface used for autostereoscopic surface generation

The implementation also allows for the user to change the focal surface used for the generation of the autostereoscopic image, which significantly impacts each sub-image in the autostereoscopic image. When the focal surface is set in front of the surface object, the display of the object in each sub-image is flipped vertically since rays from cameras at lower values on the y-axis in the camera plane map to points at larger values on the y-axis on the image plane than rays from cameras higher on the y-axis. Then, as the focal surface approaches infinity in the z-direction, each sub-image essentially looks the same since the rays from each camera approach the center of the image plane regardless of the x and y coordinates of the focal surface due to the dominance of z-coordinate in this situation.

3.4. Autostereoscopic Image Generation - CPU Style

First, we implemented our autostereoscopic image generation on the CPU. We used the given 'pumpkin' image also used in the autostereoscopic image generation in [2]. First, we kept the focal surface constant and varied the number of focal surface points. The result using sparse sampling where the focal plane is a little behind the objects and sampled at 32 X 24 is shown in figure 2. The generation of this image using the CPU implementation was very quick, happening in less than .03 seconds, plenty fast for real-time generations.

Then, we sampled the focal plane at 320 X 240, the same as the resolution of the images in the database. The output autostereoscopic image has a resolution of 5120 X 3840 pixels, which is far too large to fully display. A small portion of the output image is shown in figure 3, and a zoomed-out display of the output image is shown in figure 4. Interestingly, the zoomed-out autostereoscopic display looks the same as the view of the scene from a single camera looking at the center of the focal plane. The generation of this densely-sampled autostereoscopic image takes about 2.5 seconds to generate on the CPU, which may not be fast enough for real-time applications.

Next, we experimented with changing the focal plane of the autostereoscopic image. The result using sparse sampling of 32 X 24 with the focal plane in front of the object is shown in figure 5. Note that the object in each sub-image appears to be flipped vertically as described in section 3.3. Finally, a portion of the result using dense sampling at 320 X 240 with the focal plane way behind the object is shown in figure 6. Note that each sub-image appears to be a low-resolution display of the entire image, as the rays from each camera approach the center of the image plane at each focal surface point as described in section 3.3.

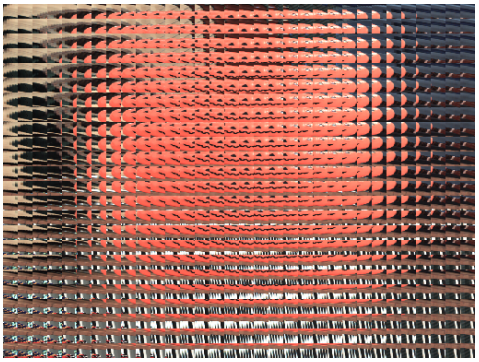

Figure 2: Generated autostereoscopic image using sparse sampling of the focal plane.

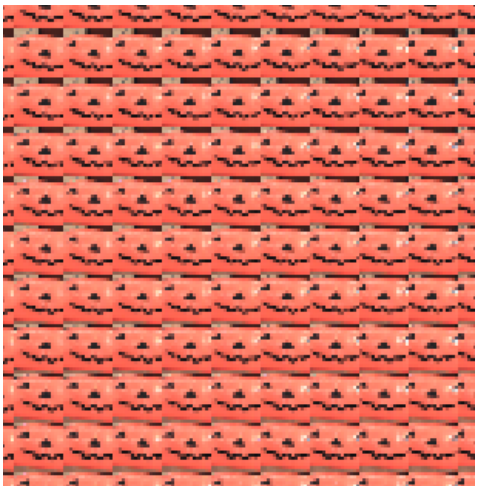

Figure 3: Portion of the generated autostereoscopic image using dense sampling of the focal plane.

3.5. Autostereoscopic Image Generation - CUDA Style

Next, we implemented autostereoscopic image generation on the GPU using the Compute Unified Device

Figure 4: Zoomed-out display of the generated autostereoscopic image using dense sampling of the focal plane.

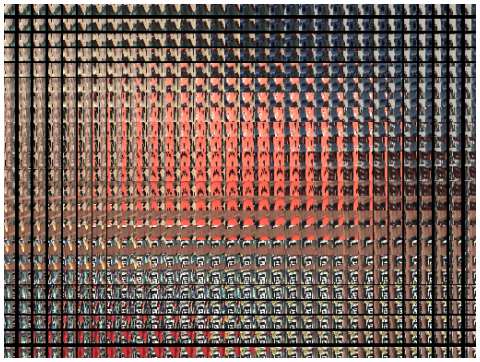

Figure 5: Generated autostereoscopic image using sparse sampling of the focal plane with the focal plane in front of the objects.

Architecture (CUDA) from nVidia in order to try and speed up the generation of the output images without losing accuracy. We will not go into a detailed description of CUDA here; the online documentation provided by nVidia provides a good introduction for the interested reader [1]. In our implementation, each of the cameras in the 16 X 16 array are mapped to a thread on the parallel architecture, and each of the points sampled are mapped to a block within the grid of thread blocks that execute in parallel. The implementation is benchmarked on two systems, a laptop with a Intel Core 2 Duo CPU running at 2.00 GHz and a nVidia 8600M GT GPU with four multiprocessors, and a desktop with a Intel Quad-core Xeon CPU running at 2.00GHz and a nVidia Tesla C870 GPU with 16 multiprocessors.

The resulting autostereoscopic images are the same in both the CPU and CUDA implementations. We will note that we were unable to produce autostereoscopic images at the sampling level of 320 X 240 on the GPU

Figure 6: Portion of the generated autostereoscopic image using dense sampling of the focal plane with the focal plane way behind of the object.

due to limitations of the CUDA architecture, but we were able to produce images at a sampling resolution of up to 306 X 210 on both implementations. We benchmarked our implementations using the 'pumpkin' image on the CPU and CUDA; the results for each system are shown in figures 7 and 8. The number of output image points represented on the x-axis of the graphs in the figures correspond to the total number of points rendered on the focal plane in the output autostereoscopic image; the number of output points using sparse sampling of 32 X 24 corresponds to a total of 768 output points. The results displayed in the figures show that the CUDA implementation is faster than the CPU implementation on both systems for both sparse sampling and dense sampling of the focal plane.

In addition, the use of CUDA for the calculations of the autostereoscopic light field image allows for more efficient visualization of the output since the output image data does not need to be transferred from the main memory to the GPU for display via OpenGL, the data can remain on the GPU due to the interoperability capabilities between CUDA and OpenGL.

4. Conclusions

In this paper, we explore the possibility of a real-time autostereoscopic image generation system using the GPU. Our results show that the generation of the desired image is faster on the GPU than in the analogous sequential CPU implementation, validating our hypothesis that the parallel architecture of the GPU can be taken advantage of for such a system.

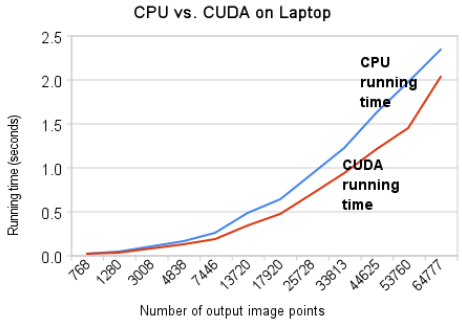

Figure 7: Running time of autostereoscopic image generation on the laptop system described in section 3.5 using the CPU and CUDA implementations.

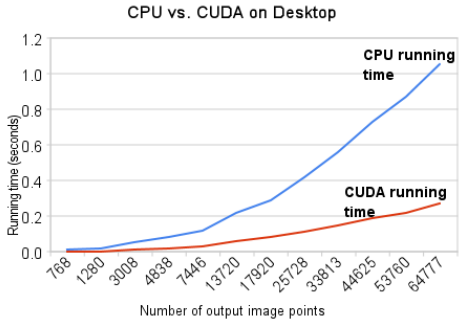

Figure 8: Running time of autostereoscopic image generation on the desktop system described in section 3.5 using the CPU and CUDA implementations.

References

[1] *NVIDIA CUDA Programming Guide: Version 2.2*. NVIDIA Corporation, April 2009.

[2] A. Isaksen, L. McMillan, and S. J. Gortler. Dynamically reparameterized light fields. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 297–306, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.