Fast computer graphics rendering for full parallax spatial displays

Michael W. Halle[a] and Adam B. Kropp[b]

[a]Department of Radiology, Brigham and Women's Hospital, Boston, MA
[b]Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA

ABSTRACT

This paper presents a computer graphics algorithm useful for rapidly generating image data for full parallax spatial displays such as full parallax holographic stereograms. Other techniques such as custom-designed ray-tracing packages and image-based rendering techniques have significant disadvantages for rapid display production. In contrast, the method described here uses scanline-based computer graphics techniques. The described implementation uses the widely available OpenGL™ graphics library and takes advantage of acceleration by computer graphics hardware subsystems. The time required to render the image data of a moderately-sized scene for a single holographic exposure is less than one second using desktop computer systems. This method is compatible with both one-step and master-transfer holographic recording geometries. Details of the algorithm are included.

Keywords: full parallax, computer graphics, holographic stereogram, holography.

1. INTRODUCTION

In the field of computer-generated holographic stereography, improvements in holographic printer design and image generation methods have recently turned full parallax displays from a rare novelty to a viable technology. A growing number of research groups are experimenting with full parallax stereograms in both the master-transfer and one-step geometries. These experiments have led to an increasing demand for an efficient yet flexible way to render image sequences for full parallax displays.

2. CHOOSING A GRAPHICS ALGORITHM

The choice of image generation algorithm always depends on the particular application. Our research, which includes automobile design previsualization, scientific and medical visualization, and a variety of other experiments, defined these preferences for our image generation method:

High speed image generation: The speed of generating images should be about as fast as it takes to expose each holographic aperture. The faster the rendering speed, the faster we can make changes and run new experiments. The ability to use existing graphics hardware is even more desirable.

Support from an external vendor: There are many more computer engineers and graphics programmers in the world than there are holographers. By using a system developed and supported by someone else, we leverage the strength of the larger computer graphics markets into the niche of holography.

Low level interface: As holographers and computer scientists, our group explores how computer graphics techniques can be used to make better holograms. To do this task best, we must be able to access as many levels of the system as possible. From this perspective, a general-purpose graphics technique is more useful than a modification to a full commercial modeling system. Low level building blocks can be used in our own systems and more easily adapted into the programs of others.

a. MWH: email: mhalle@bwh.harvard.edu. MWH was previously at the MIT Media Laboratory.
b. ABK: email: akropp@media.mit.edu

Good image quality: Designers currently use computer graphics systems to inspect models of automobile parts during the design phase. Similarly, scientists use workstation graphics for visualization. “Interactive quality” workstation graphics of the kind used to in these applications is adequate for most of our purposes, even though it is not photorealistic.

Minimal storage costs: Full parallax displays can store lots of image data in the holographic emulsion. Also storing this data on hard disks or other storage devices is prohibitively expensive in terms of equipment costs and the time required to transfer the images.

Database compatibility: Most of the databases that we use are made up of either polygonal primitives or of surface patches that can be decomposed into polygons. Database size can range from 10,000 polygons up to about 500,000. Our rendering method must be able to read data of this type and render it efficiently.

Based on these guidelines, we ruled out using a ray-tracing algorithm for our primary rendering tasks. A ray-tracer with the necessary flexibility for our needs would have to have been written and supported completely by our group, or licensed from another company. Ray-tracers are for the most part too slow to keep up with the rendering process, especially with large scene databases. Finally, the high quality of the ray-traced images is beyond our current imaging needs. We also ruled out the kinds of image-based rendering and distortion techniques that we have successfully used for horizontal parallax only displays[3]. Image storage and manipulation costs and the delay between image generation and holographic exposure were prohibitive disadvantages of image-based methods.

Instead, we have chosen an approach that relies heavily on traditional scanline computer graphics techniques as implemented in a widely available programmer’s interface, OpenGL™ [5]. The OpenGL library provides a uniform rendering interface to the full spectrum of graphics systems from video games to the most expensive and powerful graphics workstations. OpenGL is tuned to maximize rendering speed based on a particular computer’s architecture and any install graphics acceleration devices. Supported in the workstation industry and in a growing number of personal computer systems, OpenGL has the brightest future of any low-level graphics API currently available.

OpenGL does not by default provide images of the highest possible quality. Rather, it was primarily developed to support interactive computer graphics. Since OpenGL is used as a foundation upon which our collaborator’s graphics systems are built, it is perfectly matched to the needs of many of our three-dimensional imaging tasks. (When necessary, higher quality images can be rendered in OpenGL by combining several different rendered images.)

Unfortunately, graphics libraries such as OpenGL are not designed with all the needs of the computational holographer in mind. The imaging model used by almost all such systems is a “snapshot” photography mode, where a scene somewhere in the world is captured by a pinhole camera and recorded as pixels in an image. Adaptations to this model must be made to fit the optical geometry required by full parallax stereography. The next section describes this geometry and how it related to the most traditional computer graphics camera model.

3. FULL PARALLAX CAMERA GEOMETRY

Image capture or synthesis for most holographic stereograms, independent of format, follows a general rule: determine the location in space where the display images its view apertures, and place a camera at that location to capture image data for that aperture. Historically, the most common display, a two-step master-transfer stereogram where the master and its apertures are projected to the view zone, uses a camera positioned at the view zone distance. The camera moves laterally, capturing an image once for every view aperture.

In the case of a one-step holographic stereogram, the apertures are located on the holographic plate. The object usually also straddles the hologram and aperture plane in this type of stereogram. If a simple pinhole camera were used to capture images for this display, the resulting images would only contain information about the object located on one side of the plate. A camera facing away from the viewer would not, for instance, capture any information about objects in between the viewer and the plane of the plate. (We neglect the fact that a physical camera could not be used because it would collide with objects on the space.)

An optical solution to this problem for “real world” scenes and cameras is to outfit the camera with a large spherical lens. The lens moves the effective position of the aperture of the camera to a new location some distance in front of the lens. The lens is placed in the view zone, and the aperture is sequentially placed on each aperture. Such a system, in horizontal parallax only, was proposed by Benton to correct distortions in the Multiplex™ cylindrical stereogram[1]. Figure 1 shows a two-dimensional version of the modified camera.

The optical system of this camera includes the parts of the image located between the plate and the lens in addition to the rest of the scene located beyond the plate. Objects lying closer to the camera occlude those further away, and objects falling behind the camera and lens assembly are not imaged at all. This camera captures the same information about the scene we wish the holographic aperture to project when the display is illuminated.

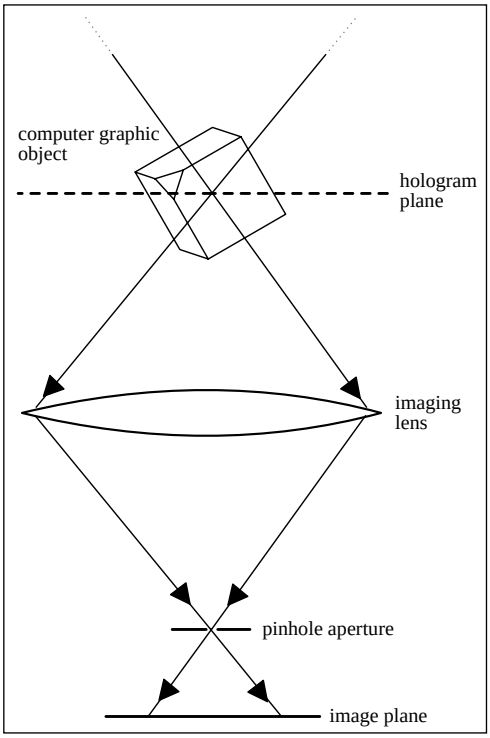

Figure 1. A large imaging lens can be used to move the effective aperture of a pinhole camera into the middle of a scene. A modified camera such as this one can correctly capture image information about parts located both in front and behind the aperture.

A computer graphics camera for full parallax displays must produce the same information as this optical camera. The camera's position must be located at the center of each aperture at the hologram plane, but the camera must also record all parts of the scene including those lying between the viewer and the hologram. The images of objects located in this region of space have several properties that distinguish them from traditional computer graphics:

- objects further from the camera (but closer to the viewer) occlude those objects closer to it,
- objects are lit and shaded based on the surface properties of the faces directed away from the camera (but towards the viewer),
- objects that are convex to the viewer are concave from the view of the camera, and vice-versa.

In effect, the image of objects located between the aperture plane and the viewer is pseudoscopic, exhibiting the same properties as an optical pseudoscopic image commonly seen in holography. The image of the rest of the scene is rendered orthoscopically.

Figure 2 shows the geometry of the computer graphics camera frustum with respect to the aperture plane and a computer graphic scene. In this case, the scene straddles the plane of a one-step full parallax hologram. The frustum of the computer graphics camera crosses itself at the center of a holographic aperture. (In the picture, the resolution of the aperture is coarse for explanatory purposes.) The image sequence is generated by changing the view transformation of the camera to move it to the center of each successive aperture.

Conventional computer graphics has little trouble generating the orthoscopic part of the rendered scene. On the other hand, rendering algorithms routinely clip away all parts of the scene behind the viewer in order to avoid confusing pseudoscopic images. For most non-holographic applications, this clipping is desirable. In this particular holographic application, though, the clip operation removes an entire half-space of the image.

The orthoscopic camera so common on computer graphics can be turned into a pseudoscopic camera by changing several of the camera's properties:

- the depth buffer should favor objects that are far from the camera, and occlude those closer to it,
- the viewport must be flipped left-to-right and up-to-down with respect to the orientation used in an orthoscopic camera,
- back face culling, an optimization used to remove backward facing surfaces of an computer graphics object, must either be disabled or configured to remove front faces instead.

Lighting is also different in the pseudoscopic world. In the Phong light model of conventional computer graphics, lighting calculations are done using three vectors: the surface orientation vector ***N,*** the vector to the eye ***E***, and the vector to each light source ***L***. The eye vector is reflected across the surface orientation vector to form an additional reflection vector ***R***. In an orthoscopic camera, the eye vector points from the surface to the camera position. In a pseudoscopic camera, though, the eye and camera are opposite each other, causing incorrect shading of the model. The flipped camera vector is denoted by ***C****.

Fortunately, OpenGL flips the surface orientation vector ***N*** for all unculled surfaces that point away from the camera to form a new vector ***N****. A mathematically-equivalent result to the calculation using ***N***, ***E***, and ***L*** can be found using ***N****, ***C****, and a new vector ***L****. Figure 3 shows this vector relationship. ***L**** is a "pseudoscopic light": a light with equal intensity and lighting properties but opposite direction, much like a converging light source in optics. OpenGL does not implement generalized con-

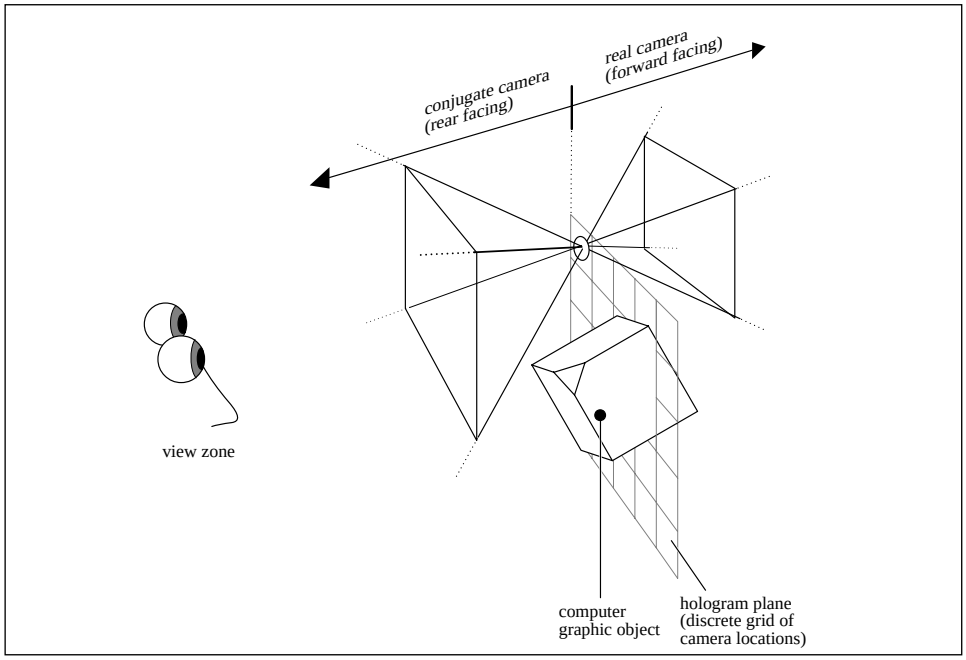

Figure 2. The frustum of a camera used to capture image data for a single holographic aperture of a full parallax display crosses itself at the aperture plane. Since common computer graphics algorithms do not allow a camera with such a frustum, the needed image must be synthesized by using two different cameras.

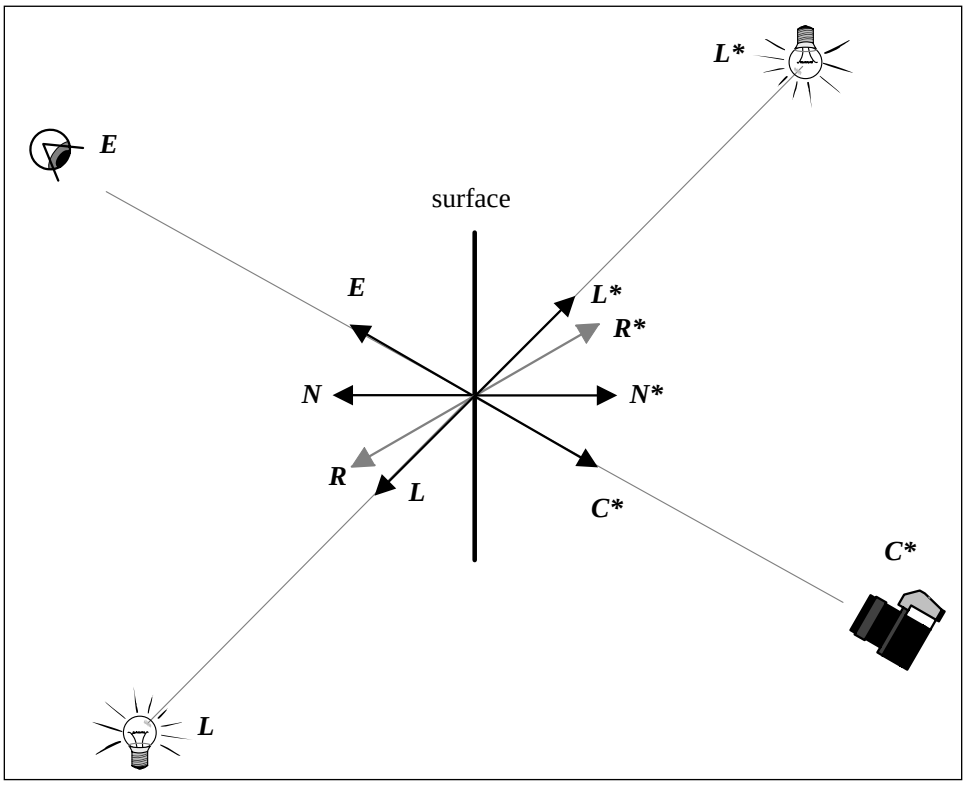

Figure 3. The standard computer graphics relationship between the eye vector ***E,*** the surface orientation vector ***N***, and the light source vector ***L*** is mathematically equivalent to that between the pseudoscopic camera's vector ***C****, the automatically-flipped normal vector ***N****, and the manually flipped light source vector **L***.

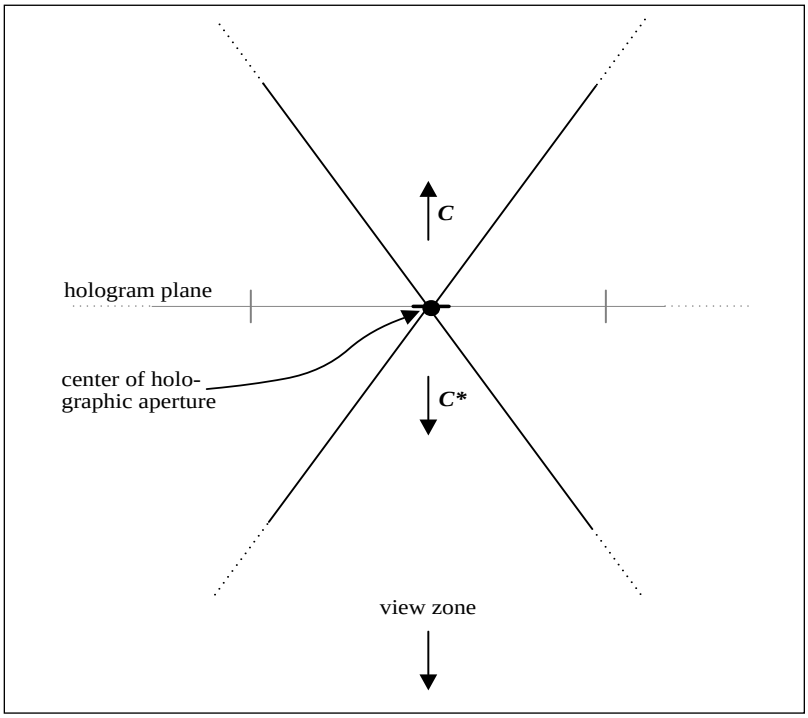

Figure 4. A two-dimensional representation of the two pieces of the double frustum camera. The orthoscopic camera is called *C*, the pseudoscopic camera is referred to as *C**.

verging light sources. However, a directional light source, with a location at infinity, is its own conjugate source and can be used for both orthoscopic and pseudoscopic rendering. Correct pseudoscopic rendering requires only that the direction of the light source be flipped.

Figure 2 can now be thought of as showing two frusta from two different cameras: an orthoscopic camera located at the plate looking away from the viewer, and a pseudoscopic camera facing back. These cameras are conjugate pairs: we will designate the orthoscopic camera as *C* and the pseudoscopic camera as *C**. A two-dimensional version of this camera geometry is shown in Figure 4. *C* and *C** image exclusive parts of space. To form a complete image of the scene, the view computed by *C* is overlaid by the view taken by *C**. Objects seen by *C** are all in front of objects seen by *C*. In computer graphics, this overlay can be done by rendering *C*'s image, changing to camera *C**, clearing the depth buffer to a value close to the camera, and rendering *C**'s image.

In short, to render the image data for a single aperture of a full parallax display, the computer graphics camera must mimic the optical behavior of that aperture. In order to render the bundle of rays that pass through an aperture, the computer graphics camera requires an unusual view frustum. Although current graphics algorithms do not directly support a camera with this kind of view, the behavior of the camera can be simulated by breaking up the frustum into smaller pieces and configuring the rendering algorithm to correctly synthesize and composite the resulting image. Although this paper gives examples based on a one-step geometry where the scene straddles the aperture plane, the same camera geometry can be used for any other aperture plane location.

4. CLIPPING PLANES

Computer graphics cameras actually have two different depth clip planes, a near plane and a far plane. The far plane sets the a limit for the most distant object in the scene. The near plane clips the objects located closest to the camera. Because of a mathematical singularity, the near plane is always located at least some distance from the camera's position. If two cameras with this offset near plane were used to produce the double frustum camera by positioning them at the aperture plane, a gap would be formed between the two near planes. Objects falling inside the gap would be clipped or disappear entirely.

To fix this problem, the near plane is placed close to the camera position for both *C* and *C**. Both cameras are centered laterally with respect to the view aperture, but they are shifted in depth so that their near planes coincide at the aperture plane.

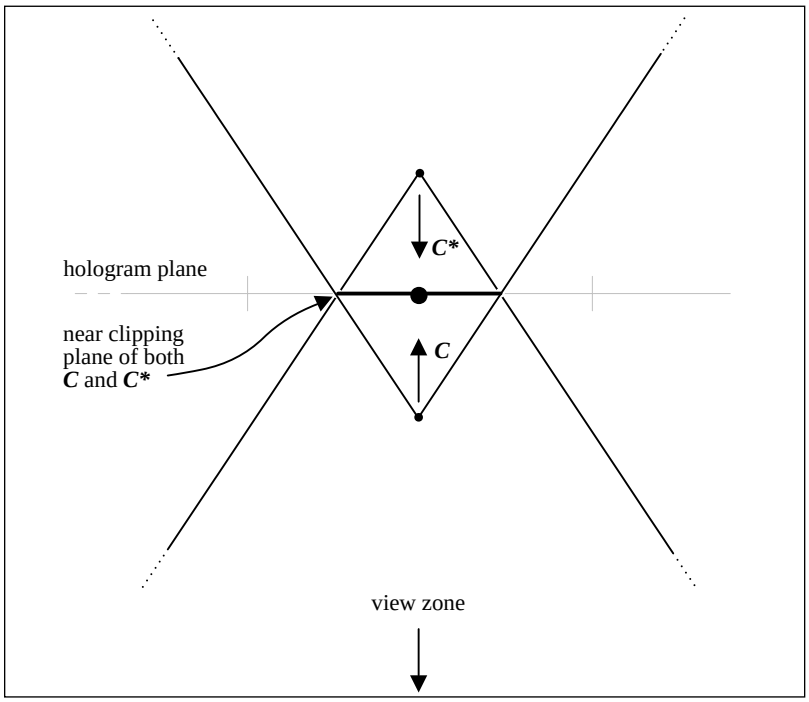

Figure 5. Computer graphics camera geometries have a near plane located a non-zero distance away from the camera position. C and C* must be positioned so that their near planes overlap at the aperture plane; otherwise, parts of the scene will be imaged twice or not at all.

Figure 5 shows the resulting geometry. In general this approach produces no noticeable image problems. Two potential problems could result. First, small image distortions could occur. Second, a near plane positioned very close to the camera can result in coarse quantization of some depth planes. If these artifacts prove to be a problem in some application, the frusta can be broken up further into smaller sections, each rendered in a separate rendering pass.

5. RECENTERING GEOMETRY

To this point, this paper has discussed a camera view frustum centered in relation to the respective view aperture. This kind of graphics camera is compatible with an exposure system where the projection screen or diffuser element used to expose each element is positioned in front of each aperture during exposure. In other situations, it may be preferable to fix the location of the projection screen with respect to the holographic plate, not individual apertures. To produce images for this system, the double frustum must be laterally sheared to recenter the plane corresponding to the projection screen when each view is generated. Shearing is centered around the aperture plane, with one side of the frustum moving to one direction and the other side moving to the other. Figure 6 shows an example of the shearing operation.

6. OTHER EXTENSIONS

The rendering algorithm described above can extended to include environmental reflection maps. Reflections of sky, interior area light sources, and the ground are important tools for automobile designers and thus an essential part of our rendering algorithm. Reflection mapping requires separate rendering passes to synthesize the diffuse and reflected image data for each camera type.

Previous work from our group[4] has demonstrated the need for proper band limiting of image detail in stereoscopic displays. Maximum resolution in all such displays is dependent on the holographic aperture size, the resolution of each aperture's projected image, and the depth of the scene detail. One technique for anti-aliasing the scene generated with the double frustum camera is to shear the frustum laterally with the projection screen plane being fixed, repeatedly sampling the entire area of the holographic aperture with the apexes of the two frusta.

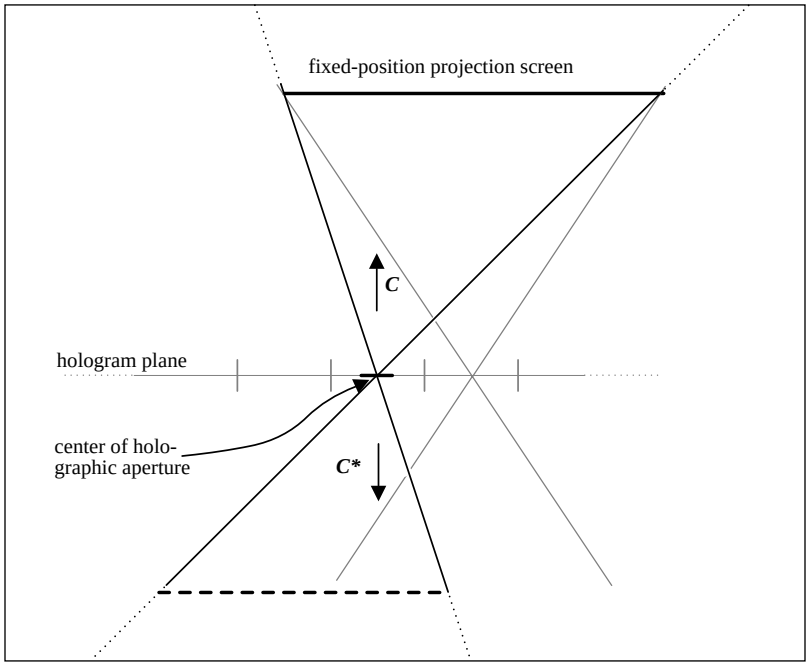

Figure 6. Holographic exposure geometries that use a projection screen fixed with respect to the hologram plane require the double frustum to be sheared to recenter the projection screen area. Cameras *C* and *C** are sheared in opposite directions with respect to the the hologram plane.

7. RESULTS

Table 1 quantifies the performance of a prototype rendering implementation of the full parallax rendering system, including reflection maps. Performance tests were run on two Silicon Graphics computers: a 150MHz R4400 Indigo2 workstation with a Maximum Impact graphics system running IRIX 5.3, and a two processor 150MHz R4400 Onyx RealityEngine2 also running IRIX 5.3. The graphics algorithms between the two machines were identical except for the fact that hardware anti-aliasing on the RealityEngine2 system was enabled while on the Maximum Impact system it was not. Figure 7 shows examples of the images generated with this algorithm.

number of polygons	% polygons reflection mapped	RE2 frames/sec	RE2 polygons/sec	Mx. Impact frames/sec	Mx. Impact polygons/sec
42,382	31.2%	1.98	84,077	3.42	144,868
272,074	47.9%	0.258	70,286	0.58	159,710

Table 1. Performance data for rendering images of two different polygon databases using the full parallax computer graphics camera.

8. CONCLUSION

By using conventional computer graphics in an unconventional way, we have developed a general purpose system for rendering image data for full parallax displays. Since the underlying graphics library is a widely-available commercial package, time that otherwise would be spent maintaining rendering algorithms can instead be filled building new systems that use them and developing new kinds of displays that benefit from them. Images can be rendered at speeds comparable to that of current holographic exposure mechanisms, permitting experiments in full-parallax imaging to be done rapidly.

ACKNOWLEDGEMENTS

The work in this paper has been funded by the Honda R&D Company, International Business Machines, and Nippon Electric Company.

REFERENCES

1. S. A. Benton, “Distortions in cyclindrical holographic stereogram images”, *J. Opt. Soc. Amer.* **68**, 1440A, Oct. 1978.

2. K. Haines and D. Haines, “Computer graphics for holography”, *IEEE Computer Graphics and Applications*, January, 1992, pp. 37-46.

3. M. W. Halle, S. A. Benton, M. A. Klug, J. S. Underkoffler, “The Ultragram: a generalized holographic stereogram”, SPIE Proceedings #1461 “Practical Holography V” (SPIE, Bellingham, WA, 1991), S. A. Benton, editor, pp. 142-155.

4. M. W. Halle, “Holographic stereograms as discrete imaging systems”, SPIE Proceedings #2176 “Practical Holography VIII”, (SPIE, Bellingham, WA, 1994), S. A. Benton, editor, pp. 73-84.

5. J. Neider, T. Davis, M. Woo, *OpenGL Programming Guide*, Addison-Wesley, 1993.

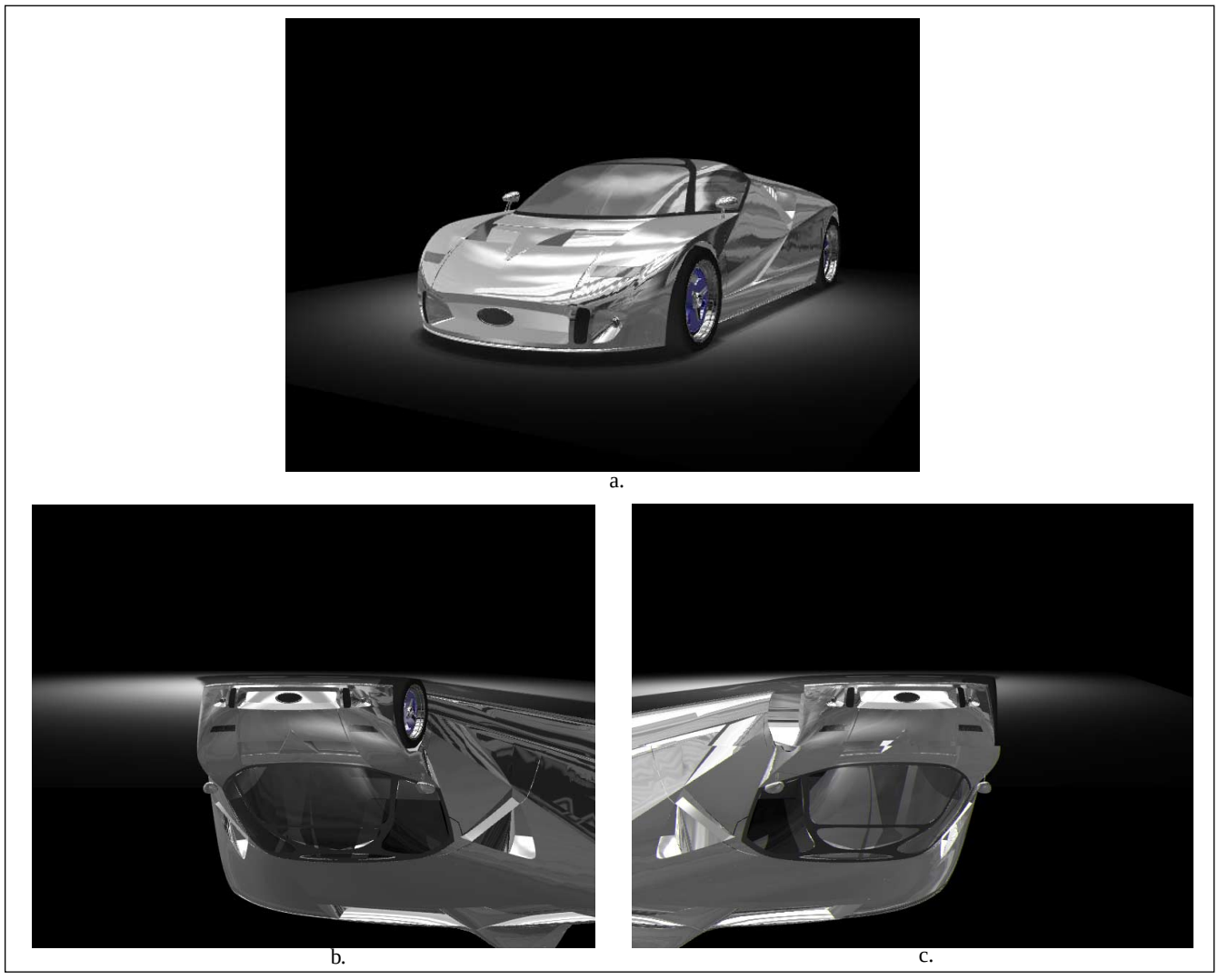

Figure 7. Three images of an automobile body scene database rendered using the full parallax camera. Figure (a) shows a view of the car captured with camera *C*. Figures (b) and (c) are different perspectives rendered using camera *C**.