Spatial Data Structures for Photorealistic and Non-photorealistic Rendering

Tokiichiro Takahashi

March 2004

Contents

1. Introduction

This thesis introduces two different kinds of spatial data structures regarding:

- Light source space (geometric relations between linear/area light sources and three dimensional object models)
- Screen space (intermediate buffers lie between 'hidden surface removal' and 'shading' processes of the standard pipeline)

These sophisticated spatial data structures have yield:

- Fast and analytic shading and shadowing algorithms with linear/area light sources, and
- New applications of computer graphics:
  - A new paradigm of 'non' photo-realistic rendering,
  - An NC machining system, and
  - A interactive photo-realistic rendering algorithm with perfect anti-aliasing,

respectively.

This chapter introduces to the importance of spatial data structures and computer graphics algorithms. A summary of our original research contributions is followed.

1.1 Data Structures for Light Source Space

To realize advanced local illumination models for linear and area light sources is one of the most important tasks in the field of photo-realistic rendering. These extended light sources cause significant visual effects such as *penumbras* (called soft shadows) and broad but sharp highlights. In fact, these visual effects are very familiar in our daily life and so notably improve image photo-realism, even if they are subtle.

There have been proposed so many methods to synthesize photorealistic images with soft shadow. They are classified into the following two major approaches. *Monte Carlo* approach needs excessive computing load, but often causes aliasing artifacts such as poor penumbras. Our approach is based on *analytic* approach which can completely suppress the aliasing artifacts since it determines unoccluded portions of each area light source exactly. This exact segmentation, termed "*light clipping*", which is expensive, is

needed for all objects in a scene at every pixel. Sophisticated spatial data structure is expected to reduce the total amount of light clipping processes.

Especially, *space subdivision* technique is simple yet powerful to handle a huge number of objects and light sources. We introduce a new sptial data structure, called *ray-oriented buffer*, which subdivides the 3D space into radial sub-sectors by following light rays radiating from polygonal light sources. Objects intersecting each sub-sector are listed in the related cell of the buffer. Thus, objects that possibly cast shadows onto an observed point can be selected by referring the buffer. The ray-oriented buffer features "*spatial coherence*".

1.2 Data Structures for Screen Space

Other important spatial data structure is regarding to intermediate buffers of the standard rendering pipeline, *i.e.* the *screen space* data structures. The standard rendering pipeline consists of *projection* (viewing conversion, perspective conversion), *hidden surface removal* including clipping, and *shading* as shown in Fig.1.1(a). Image synthesis is usually iterated until the synthesized images satisfy the user. The user may modify object layout, view point, lighting parameters, and object surface parameters inclusive of mapping. Since the iteration is mainly required to set those parameters, overall rendering time can be shortened by saving the results of hidden surface removal in a buffer. Such a buffer lies between hidden surface removal and shading operations, and store the results of hidden surface removal. However, conventional simple structured buffers cause aliasing artifacts because they digitize the object surfaces as pixels or scanlines.

In this thesis, we propose the *cross scan buffer* which preserves the results of the hidden surface removal by the cross scanline algorithm [Tanaka90] as shown in Fig.1.1(c). The cross scanline algorithm can determine the exact geometric shapes of visible surfaces. These exact shapes, *i.e.*, triangles and trapezia, are stored in the buffer. This characteristic is very useful for image re-scaling to arbitrarily size, more accurate shadow creation, exactly anti-aliased texture mapping, and so on.

Techniques for the comprehensible drawing of 3D shapes are indispensable for various applications such as industrial design or medical imaging. Their importance in

computer graphics is not at all inferior to that of photo-realistic rendering techniques. We propose the G-buffer (geometric buffer) which contains a geometric property of the visible object in each pixel [Saito90]. By extending image processing operations to G-buffer, non photorealistic rendering is achieved (See Fig.1.1(b)).

Moreover, the G-buffer method is applied to NC machining. By utilizing G-buffers created from a parallel projection, all the required NC machining functions are realized as image processing operations (Fig.1.1(b)). A total NC system is created that consists of all essential functions, such as tool path generation, path verification, and feed rate control.

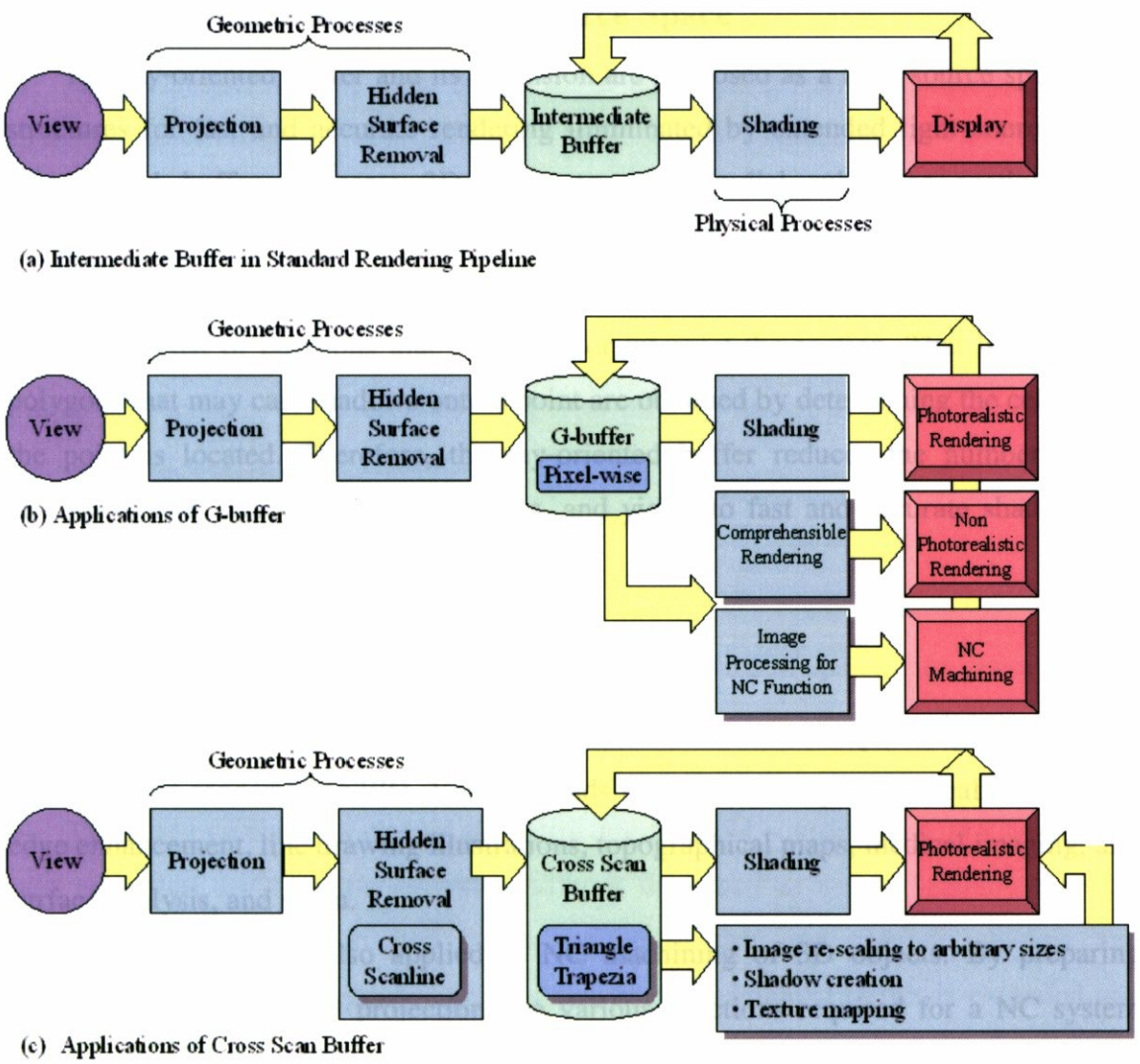

Fig.1.1 Intermediate Buffers of Rendering Pipeline and their Applications.

1.3 Contributions

The contributions of this thesis fall into two areas:

- Data structure for light source space
  - Radial space subdivision for fast and accurate rendering.
- Data structures for screen space
  - Pixel-wise subdivision both for non-photorealistic rendering and for NC machining of 3D objects.
  - Triangle/trapezia subdivision for interactive image re-generation.

1.3.1 Data Structure for Light Source Space

The ray-oriented buffer and its extension are proposed as a light source space data structures for fast and accurate rendering illuminated by extended light sources. The ray-oriented buffer segments 3D space to many radial sub-space sections. The segmentation guarantees that if a point is included in a section, all light rays falling on the point are also contained in the section. Each cell of the buffer is assigned to a section and saves a list of polygons that lie within or intersect the section. That is, candidate polygons that may cast shadows onto a point are obtained by determining the cell where the point is located. Therefore, the ray-oriented buffer reduces the number of the candidates that occlude the light sources, and yields to fast and accurate shading and shadowing.

1.3.2 Data Structures for Screen Space

The G-buffer [Saito90] pioneers a new field of computer graphics, non photorealistic rendering (NPR). Various kinds of NPR techniques are realized such as edge enhancement, line drawing illustrations, topographical maps, medical imaging, and surface analysis, and so on.

The G-buffer was also applied to NC machining of 3D objects. By preparing G-buffers from a parallel projection, the various functions required for a NC system were realized with image processing operations. This allows any surface description and any tool shape to be used.

The cross scan buffer is also proposed, which preserves accurate geometric shapes of visible surfaces in a list structure. The visible surfaces are described using the triangles and trapezia determined by the cross scanline algorithm [Tanaka90]. In addition to improvement of image re-generation speed, various kinds of applications are also possible such as image re-scaling to arbitrary sizes, two-pass shadow creation, and texture mapping.

1.3.3 Industrial Contributions

This thesis, especially the G-buffer, impacts on not only computer graphics industries but also manufacturing industries and/or industrial design fields. Thus, the G-buffer pioneers new possibilities of computer graphics technologies.

ATI Technologies, Inc., one of the major computer graphics company, has been adopting the G-buffer as its architecture [Mitchell02a], [Mitchell02b], [ATI03]. Their non photo-realistic rendering environment based on G-buffer has been widely utilized around the world. This work was done in collaboration with Takafumi Saito.

NTT has also developed a rapid prototyping system using NC milling machine for industrial design, based on the G-buffer NC machining technologies. The rapid prototyping systems were supplied to several product design and/or industrial design companies as well as some institutions.

1.4 Thesis Organization

This thesis begins with introduction to the importance of spatial data structures and computer graphics algorithms, followed by a summary of our original research contributions. We discuss related work in Chapter **2** in detail.

The next two chapters discuss data structure for light source space. In Chapter **3**, a new algorithm using the ray-oriented buffer is described as a data structure for light source space. The ray-oriented buffer realizes a fast and accurate shading and shadowing algorithm for linear light sources. The ray-oriented buffer is extended and applied to area light sources in Chapter **4**.

The next three chapters discuss data structures for screen space and their applications. Chapter **5** is about a new rendering technique with the G-buffer. They

make it possible to produce comprehensible images of 3D objects. The G-buffer is also applied to NC machining in Chapter **6**. Moreover, sophisticated intermediate buffer, the cross scan buffer, is proposed to re-generate perfect anti-aliased images in Chapter **7**.

We finish discussion, future work, and conclusions in Chapter **8**.

2 Related Work

In this chapter, related work is reviewed and compared with our achieved studies prior to the following chapters.

2.1 Fast and Analytic Rendering for Extended light Sources

Simulating various kinds of light sources is very important in generating realistic images. Accordingly, several shading and shadowing algorithms have been proposed for directional lights, point lights, spot lights, and so on. Illumination by linear and area light sources notably improves image photo-realism, since they cause penumbras along shadow boundaries. In fact, this effect, called soft shadow, is very common in our daily life. Moreover, since shadows give us good cues for recognizing object shapes, exact shadow generation is demanded. However, they are even subtle.

2.1.1 Monte Carlo Approaches

Many Monte Carlo approaches have been proposed to generate the visual effects by area lights such as distributed ray tracing [Cook84], approximation of an area light source by a cluster of point light sources [Verbeck84], Monte Carlo integration [Kajiya86], its two-pass solution [Wallace87], light-backward ray tracing [Ward94], and so on. The disadvantage of such systems is, however, excessive computing load. Reasonable computing time requires the number of sampling points to be restricted, but this often causes aliasing artifacts such as poor penumbras. This is because any point in a penumbra is illuminated by some but not all parts of an area light source.

2.1.2 Analytic Approaches

Shadowing algorithms for linear light sources must determine light segments falling on each point on objects' surfaces. The segmentation termed light clipping proceeds as follows. First, a light triangle is defined by both end points of the linear light source and the point at which illumination is being calculated. Each object is then tested to determine whether it intersects the light triangle or not. If intersection does occur, the intersection coordinates are calculated. Finally, light segments hidden by the

intersection are removed. As a result of this, the light clipping yields exact light segments, however, its cost is excessive because an expensive intersection test is needed for all objects in a scene.

2.1.2.1 Penumbra Pre-processing

Nishita and Nakamae [Nishita83] classified segments of each object of the scene as being in either umbra, penumbra, or illuminated area by using the shadow volume technique. This algorithm successfully eliminates redundant intersection tests for simple environments. However, in a common situation in which objects cast shadows onto other objects, the computing cost remains expensive because testing at each object surface involves generating a convex hull of all other objects.

2.1.2.2 Visibility Pre-processing

(a) BSP Trees

The second approach computes the visibility prior to rendering by using, for example, BSP trees [Chin92]. To reduce the cost, Bao *et al.*[Bao93] proposed a method which is an extension of the BSP shadowing algorithm for point light sources[Chin89]. However, traveling the BSP trees is also an expensive operation. In addition, processing complexity of BSP tree is $O(N^2)$ [Woo90]; here N is number of polygons in the scene.

(b) Discontinuity Meshing

Discontinuity meshing was also introduced by [Heckbert92]. That is, it is known in advance which polygons occlude light sources, thus the light clipping cost can be reduced. However, when complex objects cast shadows, excessively large partitioned meshes must be generated. In addition, both the generation and traversal of the BSP tree require high computation cost.

(c) Backprojection

The backprojection techniques [Drettakis94], [Stewart94] are complicated and still take a long time to generate discontinuity meshes.

2.1.2.3 Space sub-division

Space subdivision methods can also decrease the number of objects subjected to light clipping.

(a) 3D Grid

Uniform space subdivision, or 3D grid, is a popular technique. Each subspace, named a voxel, saves a list of objects crossing the voxel. Light rays projected toward an observed point define a light volume. The volume must be scan-converted in the 3D grid to determine intersecting voxels. Objects stored in the voxels must be gathered then unified because each object is often saved in multiple voxels. This method reduces objects forwarded to the light clipping process, however, the total cost reduction is insufficient because 3D scan-conversion is very expensive. Although space subdivision like the oct-tree is useful in reducing memory requirements, its treatment increases the total cost.

(b) Light Buffer

Cylindrical space subdivision, termed the light buffer, was proposed by Poulin and Amanatides [Poulin90]. The buffer splits 3D space to many fan-shaped subspaces along a linear light source and stores intersecting objects. Its great advantage is that the object reduction is achieved by simple buffer reference. Nether 3D scan-conversion nor object unification is needed. The one dimensional buffer was extended to yield the two dimensional variant which improved the shadowing speed as described in the previous chapter.

(c) 5D Subdivision

Arvo and Kirk [Arvo87] also proposed a 5D subdivision, each subspace of which is defined by a combination of a voxel (3D) and a solid angle (2D). It also segments the 3D space in direction to improve ray-tracing speed, however, its excessive memory requirement is not practical.

(d) Radial Scan Conversion

Let us turn to the shadowing algorithms for point light sources. Depth-buffer

shadowing [Williams78] is well known as a fast algorithm. Each cell of the buffer stores the object intersecting the light ray radiated toward the cell. This means the buffer subdivides 3D space by light ray directions. Max [Max86] proposed a unique rendering algorithm for a point light or a directional light. The algorithm scans images along the light rays from the source to execute hidden surface removal and shadow volume generation in one step. These algorithms indicate that scanning toward ray directions is very effective in shadow generation.

As a matter of course, our shadowing algorithm uses a 2D ray-oriented buffer. The next section estimates the performance of conventional space subdivision methods. Our ray-oriented buffer will then be explained with buffer generation and shadowing algorithms. Finally, performance of our algorithm will be determined.

The related works are summarized at Figure 2.1.

Approach		Techniques		Sub-division Targets	Sub-divided Target Shapes	Algorithmic Complexity	Computing Cost	References
Geometry	Analytic Solutions	Penumbra Pre-processing		Polygonal Objects	Polygons sub-divided along shadow boundaries	△	◎	[Nishita85]
		Visibility	Discontinuity Mesh + Back Projection			×	○	[Heckbert92], [Drettakis94],[Stewart94]
			BSP			×	○	[Chin92][Bao93]
		Space Sub Division	Uniform 3D	3D Space	Cartesian Uniform Sub-Space	◎	×	[Glassner86], [Fujimoto86]
			5D		Prisms	△	△	[Arvo87]
			• Light Buffer		Radial Sub-Sectors	○ (only for linear)		[Max86], [Poulin90]
			• Ray-oriented Buffer			○	◎	[Tanaka95] [Tanaka97]
Monte Carlo	Approximated Solutions	Distributed Ray Tracing + Photon Mapping	• A lot of rays are traced or a huge amount of photons are emitted to approximate a linear/an area light source. • Computation cost vs. image quality (penumbra/soft shadow)			○	×	[Cook84], [Verbeck84], [Kajiya86], [Wallace87], [Ward94]
		Approximation by Point Light Source Cluster	• A linear/an area light source is approximated by a set of point light sources. • Computation cost vs. image quality (sometimes causes moiré artifacts)			◎	△	[Bao93]

Fig.2.1 Comparison with Spatial Data Structures for Extended light Sources.

2.2 Intermediate Buffers for Advanced Applications

The standard rendering pipeline consists of the following operations [Foley90]; viewing conversion, perspective conversion, clipping, hidden surface removal, and shading. Image synthesis is usually iterated until the synthesized images satisfy the user. The user may modify object layout, view point, lighting parameters, and object surface parameters inclusive of mapping. Since the iteration is mainly required to set those parameters, overall rendering time can be shortened by saving the results of hidden surface removal in a buffer. Such a buffer would improve not only the interactive environment for image synthesis, but also new applications such as non-photorealistic rendering, or NC machining employing the rendering results.

2.2.1 Fast Image Re-generation

Several rendering buffers have been proposed including the span buffer [Whitted81], [Nakamae89], ray-tree, and G-buffers [Saito90]. They lie between hidden surface removal and shading operations, and store the results of hidden surface removal. However, conventional buffers cause aliasing artifacts because they employ scanline, ray-tracing, or z-buffer, and so digitize the object surfaces as pixels or scanlines.

2.2.2 Comprehensible Rendering

Techniques for the comprehensible drawing of 3 dimensional shapes are indispensable for various applications such as industrial design or medical imaging. Their importance in computer graphics is not at all inferior to that of photo-realistic rendering techniques. Comprehensibility is mainly created through suitable enhancement lather than by accurately simulating optical phenomena. For shape comprehension, line drawings are effectively used as an addition to or substitute for surface coloring and shading [Kondo88]. For example, profiles and edges can be enhanced with black or white border lines. Curved surfaces can be made more comprehensible by hatching with curved lines. These techniques are commonly used in hand drawn illustrations. However, they have not been adequately developed for computer graphics compared to photo-realistic rendering techniques.

The major problem for synthesizing a comprehensible image is determining the most suitable combination of enhancement techniques. The reason is that comprehensibility depends on the object, purpose, and sometimes the viewers' preferences, and cannot be expressed with theoretical definitions. Therefore, we must find the best combination by trial and error for each object or application. In order to maintain high productivity, graphics systems must be flexible and interactive to match the users' experimentation. For photo-realistic rendering, there is a lot of excellent research that aims to reduce image re-computation cost by preserving intermediate information [Duff85],[Mammem89],[Nakamae89],[Perlin85],[Porter84], and/or to build a rendering system flexibly by separating it into small procedures which can be combined freely [Cook84],[Crow84],[Nadas87],[Whitted82]. These techniques might appear to be effective for comprehensible rendering. However, enhancement using line drawings and conventional surface rendering are so different that it is difficult to combine them efficiently. This difficulty arises, for example, when eliminating hidden lines and surfaces for the same image [Saito89]. Perlin has also proposed Pixel Stream Editor [Perlin85] useful for photo-realistic rendering.

2.2.3 NC Machining

While the thrust of modern computer graphics is the visualization of the real world, people must use actual 3D objects to survive. One interface between conceptual and actual objects is that created by CAD/CAM systems coupled to numerically controlled (NC) milling machines. Unfortunately, this interface is not as efficient or productive as it should be. The problem lies in the large number of factors that must be considered when machining a complex object.

A number of sophisticated methods have been developed to address one or more of these factors, and some have turned into commercial systems. Each method can be characterized as one of two types: tool path generation, or machining simulation/verification. In path generation, the main purpose is to obtain an adequate tool path that produces an accurate shape. The tool path can be obtained from the offset surface, i.e. the trace of the limit position of the tool center, and a lot of research has been performed to calculate an accurate offset surface [Zhang86], [Sakuta87].

Kishinami *et al.* have proposed a flexible algorithm called the Inverse Offset Method [Kishinami87] which uses the rasterization technique. In simulation and verification, the tool path is verified for each factor listed above. Many systems have been developed for this purpose, and one of the major differences among them is the shape representation for interference calculation. Wang *et al.* [Wang86a], [Wang86b], Hook [Hook86], and Atherton *et al.* [Atherton87] all used a projection from a view point and applied a variation of the z-buffer algorithm. Thus, their methods are termed 'view based methods'. Kawashima *et al.*. [Kawashima89] implemented such a method by using an oct-tree data structure. Chappel [Chappel83] and Jerard *et al.*. [Drysdale87], [Jerard89a], [Jerard89b] represented the shape with 'direction vectors' from discrete points distributed on the object's surfaces.

Unfortunately, in conventional CAD/CAM systems, these two activities, tool path generation and simulation/verification, are usually independent of each other and based on different methodologies. This has two disadvantages. First, the entire software package is huge and excessively complicated. This is because different programs are required for each activity in order to accommodate various shapes and tool types. Second, it is difficult to generate tool paths by using simulation results. This function is necessary because, for example, adequacy of a tool path for fine cutting depends on the result of rough cutting.

The previous works are summarized at Figure 2.2.

Data Structures		Techniques	Applications					References
			Fast hidden surface removal	Fast shadowing	Arbitrarily re-scaling a rendered image	Non-Photo-realistic rendering	NC Machining from rendering results	
Pixel		Pixels	△					[Duff84] [Cook84][Crow84] [Porter84][Perlin85][Mam mem89]
		Shadow Buffer		○				[Crow77] [Weghorst84] [Reeves87][Salesion90]
		Ray Tree	○	○				[Murakami90]
		G-buffer	○			◎	◎	[Saito90] [Saito91]
Scanline		Multi-scan Buffer	○					[Whitted82][Nakamae89]
Area	Quad Tree	BSP	△	○				
	Polygon	Cross scan buffer	○	◎	◎			[Tanaka94]

Fig.2.2 Comparison with Intermediate Buffer Data Structures for Advanced Applications.