NAVIGATION AND ILLUMINATION CONTROL FOR IMAGE-BASED VR

Tien-Tsin Wong
Hong Kong University of Science & Technology

Chi-Wing Fu and Pheng-Ann Heng
The Chinese University of Hong Kong

ABSTRACT

Simulation sickness in virtual reality applications is usually due to the non-realtime display of virtual environment. Realtime display of arbitrarily complex scene is a hard problem in traditional geometry-based computer graphics. Image-based modeling and rendering (IBMR) provides an alternative approach whose rendering time complexity is independent of scene complexity. However, due to the lack of geometrical information, capabilities that are obvious to geometry-based virtual reality become difficult problems in image-based virtual reality. In this paper we discuss how two fundamental capabilities, navigation and illumination control, can be achieved in image-based virtual reality applications.

To navigate an image-based scene, we need to know where pixels should be moved to and how to solve their visibility. While correspondences or optical flows can answer the former question, the visibility is more difficult to answer, as depth information may not be available. Deriving from epipolar geometry, we propose a triangle-based visibility-ordering algorithm, which can correctly resolve the occlusion without depth information. To control the illumination, we propose a new image representation that not just allows navigation but also re-rendering under various illuminations. By treating each image pixel as an ordinary surface element, we measure the apparent BRDF of each pixel from reference images. By manipulating these apparent pixel BRDFs, we are able to re-render (change the illumination of the scene in an image) without any geometry information. Even shadows can be correctly re-rendered.

1. Introduction

The inability of realtime rendering of arbitrarily complex scene is one of the major causes of simulation sickness in virtual reality applications. Traditional geometry-based computer graphics requires significant amount of time to render complex scenery due to the dependency of the rendering time on scene complexity. Even with the state-of-the-art graphics accelerator, realtime rendering is still far from satisfactory. Image-based computer graphics provides an alternative to render complex scene within a short period of time. It is especially useful in virtual reality applications to prevent simulation sickness. Several image-based approaches [15, 12, 8, 18] have been proposed in recent years. One well-known example of using image-based computer graphics in virtual reality is Apple's *QuickTime VR* [3]. However, due to the lack of geometrical information, capabilities that are obvious in geometry-based computer graphics, such as navigation and illumination control, become difficult in image-based computer graphics. In this paper we describe how navigation and illumination control in image-based computer graphics can be solved without knowing the geometry of the scene.

To navigate (i.e. generate an image from a new viewpoint) in the virtual environment represented by a set of reference images, there are two sub-problems to solve. The first is where pixels should be moved. It can be solved by pixel reprojection [4] if depth is known or by correspondence determination [6] if depth is unknown. The second sub-problem is the visibility problem: of determining which pixel is in the front if multiple pixels move to the same position in the new image. The most straightforward method is depth buffering. However, in some cases depth information may not be available or accurate. This is especially common for real-world photographs.

In that case, only correspondences or optical flow information can be determined, and visibility cannot be solved by depth buffering. McMillan [15, 14] proposed a clever solution to this visibility problem. The visibility is solved by drawing pixels in a specific order. Mark *et al*. [13] and Shade *et al*. [19] applied this pixel-based drawing order to resolve visibility in their works. Unfortunately, the algorithm is time-consuming, as it can only draw one pixel at a time. It would be more efficient if neighboring pixels can be grouped together and drawn at once using existing graphics hardware. However, McMillan's algorithm does not apply to image entities larger than a pixel. In this paper we extend this pixel drawing order to triangles in order to accelerate the rendering.

Another important capability of traditional computer graphics is illumination control. Without geometry, changing the illumination is no longer obvious. Standard illumination models, such as Phong's model, are no longer applicable to image-based computer graphics. Several attempts have been made to re-render the image without knowing the geometry. Nimeroff *et al*. [16] re-rendered the scene under various natural illumination (overcast or clear skylight) with the knowledge of the empirical formula that model the skylight. Belhumeur and Kriegman [1] determined the basis images of an object with the assumptions that the object is convex, and all surfaces are Lambertian. The image can then be re-rendered using linear combination of these basis images. However, the illumination is uncontrollable, as the coefficients used in the linear combination are not directly related to the direction of light source. In the second half of this paper we describe a new image-based BRDF representation, which not only allows the change of viewpoint, but also the change of illumination. There is no restriction on the shape and surface properties of scene objects as in previous approaches. Both indoor and outdoor illumination can be synthesized. Most important, the illumination is controllable.

Throughout this paper we concentrate on discussing the visibility and the illumination control of planar perspective images. For virtual reality applications, panorama is more appropriate. The discussion below can be trivially extended to cube-based panorama represented by six planar perspective images. The same algorithm can be applied on each of the six faces of a cube-based panorama.

2. Navigation

As mentioned earlier, navigation can be broken into two sub-problems. The sub-problem of moving pixels can be found in several previous works [4, 11]. In this section we focus on the second sub-problem, the visibility problem.

2.1 EPIPOLAR GEOMETRY

Before describing the proposed algorithm, we first describe some basics of epipolar geometry. Consider a planar perspective image I_c captured with the center of projection at $\dot{c}$. We use the overhead dot notation $\dot{a}$ to denote a 3D point and the overhead arrow notation $\vec{a}$ to denote a 3D directional vector. A desired image I_e is generated with a new center of projection at $\dot{e}$. Figure 1 shows the geometry in 3D.

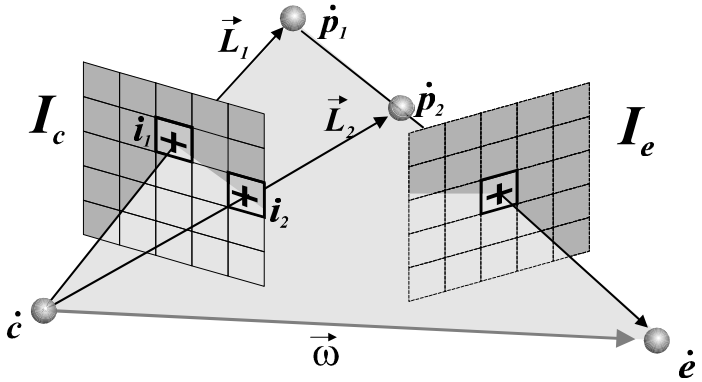


Figure 1: The geometry of two cameras in 3D.

Each pixel i in the image I_c stores the radiance along the ray $\vec{L}$ which is fired from $\dot{c}$ passing through the pixel window associated with i. Next let us choose an arbitrary pixel i_1 from image I_c. A ray $\vec{L}_1$ is associated with it. The intersection point $\dot{p}_1$ associated with i_1 must lie somewhere on the ray $\vec{L}_1$. To generate a new view from $\dot{e}$, $\dot{p}_1$ has to be reprojected onto I_e. The plane constructed by $\dot{c}$, $\dot{e}$, and $\dot{p}_1$ is known as the *epipolar plane* in computer vision literature. The vector, $\vec{w}$, originating from $\dot{c}$ and pointing towards $\dot{e}$, is called the *positive epipolar ray* while the vector, $-\vec{w}$, originating from $\dot{c}$ and pointing in the opposite direction, is called *negative epipolar ray*.

Now let's choose another pixel i_2 from image I_c. Occlusion occurs only when $\dot{p}_1$ and $\dot{p}_2$ are reprojected onto the same 2D location in I_e (see Figure 1). If $\dot{p}_2$ does not lie on the epipolar plane associated with $\dot{p}_1$, then $\dot{p}_1$ and $\dot{p}_2$ will never occlude each other. Hence occlusion occurs only when $\dot{c}$, $\dot{e}$, $\dot{p}_1$, and $\dot{p}_2$ all lie on the same plane. Moreover, the necessary condition for $\dot{p}_2$ occluding $\dot{p}_1$ is that $\dot{e}$, $\dot{p}_1$ and $\dot{p}_2$ are collinear and $\dot{p}_2$ is between $\dot{p}_1$ and $\dot{e}$, as illustrated in Figure 1.

From Figure 1 we know that $\dot{p}_2$ will never be occluded by $\dot{p}_1$ as viewed from $\dot{e}$ no matter where the exact positions of $\dot{p}_1$ and $\dot{p}_2$ might be. Therefore if we always draw i_1 before i_2 during reprojection, the visibility problem is solved without knowing or comparing their depth values. Hence if we can identify those pixels whose intersection points may occlude each other and derive the drawing order, the visibility problem can be solved without depth buffering.

To identify the pixels that can occlude each other, we first intersect the epipolar plane with the planar projection manifold (image I_c). The intersection line is called the *epipolar line*. Figure 2 illustrates the terminologies graphically. When the positive epipolar ray $\vec{w}$ intersects with the projection manifold I_c, the intersection point on the projection manifold is known as the *positive epipole*. Figure 2 denotes it by a positive sign. On the other hand, if the negative epipolar ray intersects with the projection manifold, the intersection point is known as the *negative epipole* and is denoted by a negative sign. Note that all epipolar lines pass through the epipole (either positive or negative). When the epipolar rays are parallel to the planar projection manifold, no intersection point is found on the plane. All epipolar lines are in parallel.

All pixels in I_c that lie on the same epipolar line have a chance to occlude each other. Figure 3 shows two pixels, i_1 and i_2, lying on the same epipolar line. Their associated intersection points $\dot{p}_1$ and $\dot{p}_2$ are coplanar and may occlude each other. And $\dot{p}_1$ will never occlude $\dot{p}_2$ as $\dot{p}_1$'s angle of derivation q_1 is greater than q_2 of $\dot{p}_2$. In other words, if i_2 is closer to the positive epipole on the epipolar line than i_1, then i_1 will never occlude i_2. Hence we should always draw i_1 first. The arrow on the epipolar line in Figure 3 indicates the drawing order of pixels. On the other hand, if i_2 is closer to the negative epipole on the epipolar line than i_1, then i_2 will never occlude i_1. By intersecting all of the epipolar planes with the image I_c (Figure 2), we obtain pictures of drawing order (Figure 4). Note that once $\dot{c}$ and $\dot{e}$ are known, the picture of drawing order is already determined. It is not necessary to define the epipolar planes explicitly. Hence no depth information is required in constructing the drawing order.

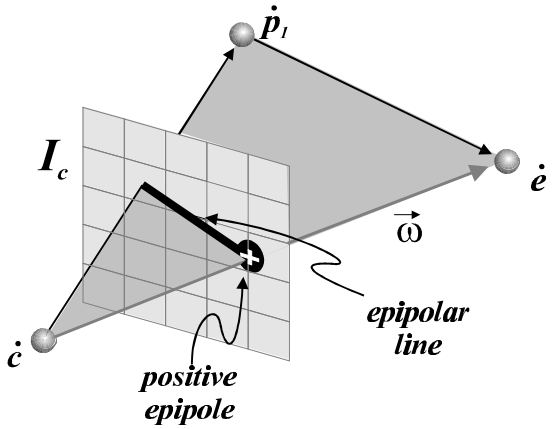


Figure 2. The epipolar line is the intersection of the projection manifold and the epipolar plane.

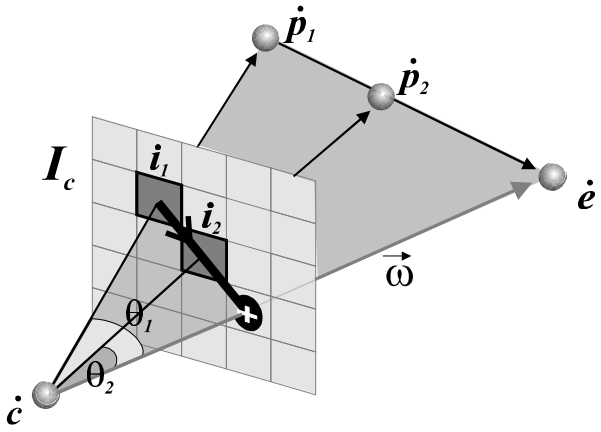
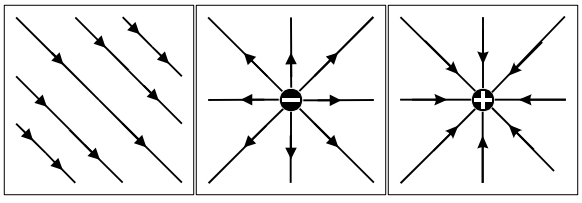


Figure 3. The drawing order between two pixels that lie on the same epipolar line.

Only three main categories of drawing order exist. If the positive epipolar ray intersects the projection manifold, a converging pattern will be obtained. (Figure 4(a)) On the other hand, if the negative epipolar ray intersects the projection manifold, a diverging pattern will result. (Figure 4(b)) If the epipolar rays are parallel to the projection manifold, the epipoles can be regarded as located infinitely far away and the epipolar lines will be all in parallel. (Figure 4(c)) McMillan [15, 14] used these drawing patterns to solve visibility without depth buffering.



(a) (b) (c)

Figure 4. The drawing patterns.

Pixels on different epipolar lines can be drawn in arbitrary order. However, the epipolar lines only tell us the ordering of pixel-sized entities that lie on the same line. If we group pixels to form larger entities (such as triangles) that overlap with multiple epipolar lines (Figure 5), the ordering of them is not clear.

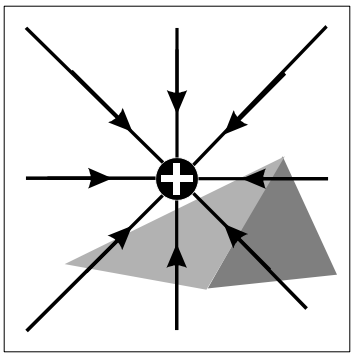
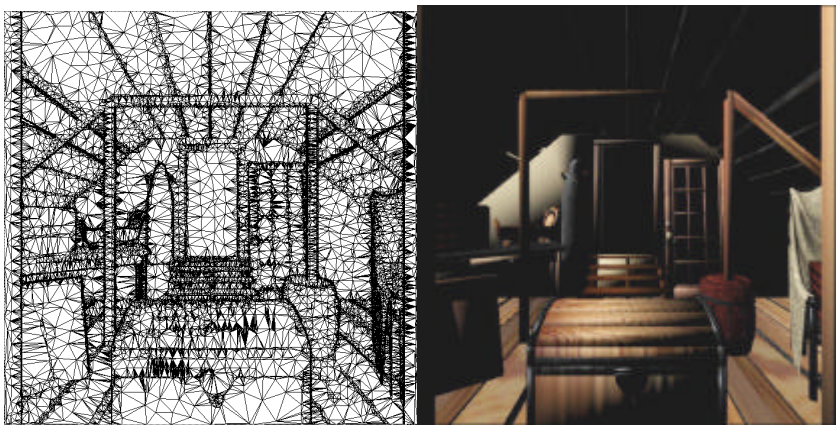


Figure 5. Larger image entities overlap with multiple epipolar lines.

2.2 TRIANGULATION

To warp an image efficiently with triangles, we first have to form the reference image into a set of triangles based on the associated depth map or the map of optical flow. The basic idea is to group adjacent pixels with similar depth or optical flow. Since the depth has a scalar value, the depth map can be regarded as a special kind of heightfield. Several algorithms [17, 5] have been proposed to "triangulate" the heightfield. If only the optical flow is available, the magnitude of the gradient of optical flow can also be used for triangulation. Figure 6 shows the result of triangulation.



(a) *(b)*

Figure 6. Triangulation of the attic scene.

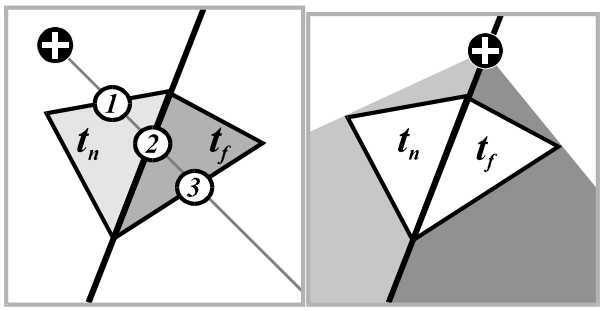
2.3 ORDERING OF NEIGHBORING TRIANGULES

Given two arbitrary triangles, t_1 and t_2, obtained by triangulating the image, we can check whether these two triangles may occlude each other by checking the range of epipolar lines the triangles occupy. If elements (pixels) in these two triangles share any common epipolar line, the ordering of these two triangles is relevant. On the other hand, if two triangles do not share any common epipolar line, the ordering of the triangles is irrelevant.

Instead of considering the order of any two arbitrary triangles from the mesh, we first consider the ordering between each pair of neighboring triangles that share a common edge, as in Figure 7(a). Now we shall show that the ordering of any two neighboring triangles can be determined by the position of the positive or negative epipole.

Theorem 1 *Given two neighboring triangles that are sharing a common edge, the planar projection manifold can be divided into two halves by extending the shared edge. The triangle with the positive epipole on its side should be drawn later during warping. On the other hand, the triangle with the negative epipole on its side should be drawn first during warping. If the epipole (either positive or negative) lies exactly on the shared edge, the ordering of these two triangles will be irrelevant.*

Proof: Let's denote the triangle with the positive (negative) epipole on its side as t_n and the other as t_f. All epipolar lines on the planar projection manifold must be straight lines, and they must all pass through the positive (negative) epipole. Now we can draw a straight epipolar line starting from the positive (negative) epipole and passing through both t_n and t_f. Since the positive (negative) epipole is on the same side as t_n whenever the straight epipolar line passes through both t_n and t_f, it should first pass through t_n, followed by the shared edge and finally t_f. (Figure 7(a)) Therefore whenever there are elements in t_n which are sharing a common epipolar line with some elements in t_f, the elements in t_n should be closer to the epipole than those in t_f. If the epipole is positive, all elements in t_n are closer to the positive epipole than any element in t_f. Hence no element in triangle t_f will occlude any element in t_n, and we must draw t_f before t_n during warping, denoted as $t_f \rightarrow t_n$. On the other hand, if the epipole is negative, no element in t_n will occlude any element in t_f as all elements in t_f are farther away from the negative epipole. Then we must draw t_n before t_f during warping, denoted by $t_n \rightarrow t_f$.



(a) *(b)*

Figure 7. (a) The straight line starting from the epipole always enter t_n before t_f whenever the line cuts both t_n and t_f. (b) If the epipole lies on the shared edge, the epipolar bands of these two neighboring triangles have no intersection.

When the epipole lies on the shared edge (Figure 7(b)), one can always separate these two triangles by drawing a line, which coincides with the shared edge. In other words, no element in t_n and t_f shares any common epipolar line. It follows that their ordering is irrelevant and we denote this relationship as $t_n \leftrightarrow t_f$.

If the epipolar ray does not intersect with the planar projection manifold, the epipoles can be regarded as located infinitely far away. All epipolar lines are then parallel and pointing from the negative epipole to the positive epipole. We can still determine which triangle is on the same side as the infinite epipole by determining the direction of the epipolar line.

2.4 TOPOLOGICAL SORTING

Using the simple method described, one can always derive the drawing order of two neighboring triangles. This ordering can be further extended to cover any two arbitrary triangles from the mesh by constructing a drawing order graph. By representing each triangle as a node and the relation $\rightarrow$ as a directed edge in the graph, we can construct a graph of drawing order. No edge is needed to represent the relation $\leftrightarrow$ as the ordering is irrelevant. Note that the constructed graph may contain disjointed subgraphs. Figure 8(a) shows seven connected triangles. The drawing order of each pair of neighboring triangles are shown as arrows crossing the shared edges between neighboring triangles. The constructed graph is shown in Figure 8(b). Figure 8(c) shows two valid drawing orders derived from the example graph. Note that there is no unique ordering for the same graph.

There is no need to construct the graph explicitly. The graph can be implicitly represented as a set of ordering relations between each pair of neighboring triangles. Hence for each shared edge we determine the drawing order between neighboring triangles using Theorem 1. The time complexity of the graph construction is obviously *O(E)* where *E* is the number of shared edges. As each triangle has three edges, *E* must be smaller than *3N* where *N* is total number of the triangles. Hence, the time complexity should be linear to the total number of triangles.

The final step to discover the ordering of all triangles is to perform a topological sort on the drawing order graph. The details of topological sort can be found in various introductory algorithm literatures [10, 21]. The basic idea of topological sort is to output one triangle t_1 at a time such that no other triangle is needed to be drawn before t_1, i.e. t_1 is not on the right hand side of any $\rightarrow$ relation. The time complexity of the topological sort is *O(E+N)* where *E* is the number of relations (edges in the graph) and *N* is the number of triangles. Since *E* is at most *3N*, the time complexity is actually linear to the number of triangles.

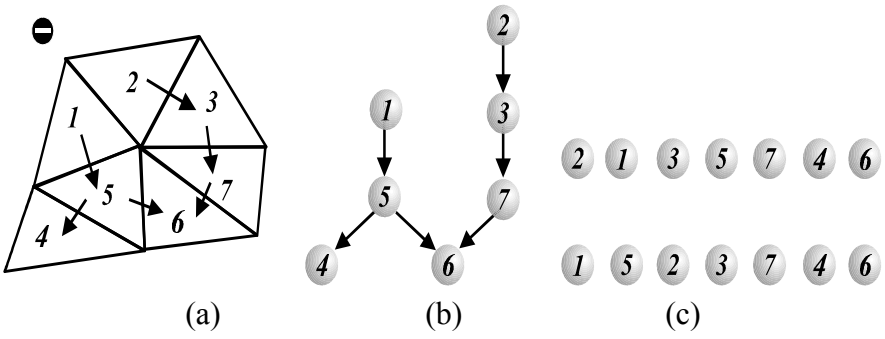


Figure 8. Construction of drawing order graph.

		Triangle-based Warping			Pixel-based Warping		
	Image	Number of	Visibility	Render	Number of	Render	Reduction
Data set	Resolution	triangles	sorting (sec.)	(sec.)	pixels	(sec.)	in %
attic	512 x 512	18582	0.060	0.020	262144	0.399	79.95%
Beethoven	512 x 512	14969	0.048	0.016	262144	0.399	83.95%

Table 1. Comparison of rendering time between triangle-based and pixel-based warping

2.5 EVALUATION

Figure 13 shows three frames from an animation sequence of warping an image of a Beethoven statue. The images on the first row show the result if they are forwardly warped in a pixel-by-pixel manner. Since no splatting is performed, gaps exist in between the pixels. Images on the second row are the final images after running our algorithm. The third row shows the corresponding warped triangulations, together with the drawing order. To distinguish one triangle from another, we use three distinct colors to distinguish neighboring triangles. The intensity of triangle indicates the drawing order. The darker the color, the earlier the triangle in the drawing order is. Figures 13(h) and 13(i) show how the visibility is resolved by the drawing order. Note that all triangles in the front (those closer to the viewpoint) are lighter in color than those in the back. The coloring in Figures 12(d) and 13(g) look random because the drawing order is irrelevant. In this case, all triangles are visible, and no occlusion occurs as viewed from the original viewpoint.

An indoor scene is shown in Figure 12. The first row shows the final warped images while the second row shows the warped triangles together with the drawing order. Note how the visibility is correctly resolved even though no depth buffering is used. The shaft in the attic scene correctly overlaps the background without comparing depth values. Since the image is now triangulated as a set of connected triangles, no gap exists between them. The holes between the triangles in our result are intentionally introduced to prevent excessive elongation after warping two neighboring triangles with discontinuous depth or optical flow values.

We have compared the speed of triangle-based and pixel-based warping on an SGI Octane computer with MIPS R10000 CPU and an MXE graphics engine. When graphics hardware is utilized, Table 1 shows that the triangle-based warping has a significant improvement in term of rendering speed over the pixel-based warping even though the determination of drawing order is purely done by software. The column “Visibility sorting” indicates the average time needed for the image-based visibility sorting (the determination of drawing order). The column “Render” indicates the average time needed for drawing triangles onto screen, which is done by the graphics accelerator. As expected, visibility sorting takes a longer time as it is done purely by software. Nevertheless, there is still a significant reduction in the total rendering time as compared with pixel-based warping. The rendering time is reduced at least 80% in all test cases. Obviously the speedup is strongly related to the number of triangles. Since the resolution of the two test scenes is the same, the frame rate of pixel-based warping is also the same in both test cases.

3. Illumination Control

Another important capability in virtual reality applications is illumination control. The change of illumination not only improves realism, but also recognition of objects in the virtual environment. Unfortunately, once the scene has been captured and represented as a set of reference images, we cannot apply standard illumination models to change the illumination because no geometry information is available. In this section we propose a new image representation to allow the inclusion of illumination.

3.1 BRDF OF PIXEL

The *bi-directional reflectance distribution function* (BRDF) [9] is the most general form representing surface reflectivity. To calculate the radiance emanating from a surface element in a specific direction, the BRDF of this surface element must first be determined. Methods for measuring and modeling the BRDF can be found in various sources [2, 20]. The most straightforward approach that includes the illumination variability of image-based rendering system is to measure the BRDF of each object material visible in the image. However, measuring BRDFs of all objects in a real scene is tedious and often infeasible. Imagine a scene containing thousands of small stones, each with its own BRDF. The situation is even worse when a single object exhibits spatial variability of surface properties. Furthermore, associating a BRDF with each object in the scene implies that the rendering time has to be dependent on the scene complexity.

Our solution is to treat each *pixel* on the image plane as a surface element with an *apparent* BRDF. Imagine the image plane as an ordinary planar surface, and each pixel can be regarded as a surface element. Each surface element emits different amount of radiant energy in different direction under different illumination. In order to measure the (apparent) BRDF of each pixel, the location of image plane must be specified (see Figure 9), not just the viewing direction. By recording the BRDF of each pixel (Figure 9), we capture the aggregate reflectance of objects visible through that pixel window. The light vector *L* from the light source and the viewing vector *V* from the view point *E* define the two directions of BRDF. This approach does not depend on the scene complexity. Moreover, it is also a unified approach for both virtual and real world scenes.

Note that the apparent BRDF represents the response of object(s) to light in each direction *in the presence of the rest of the scene*, not merely the surface reflectivity. If we work from views (natural or rendered) that include shadows, it follows that shadows appear in the reconstruction.

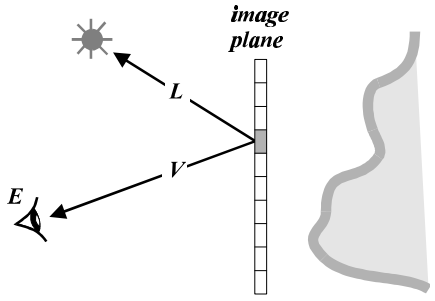


Figure 9. Measuring the BRDF of the pixel.

3.2 MEASURING BRDF

To measure the BRDF, we have to capture the images of virtual or real world scene under different illuminations. A directional light source is cast on the scene from different directions. Rendered images and photos of virtual or real world scene are captured as usual. The algorithm is

For each view point E
For each directional source's direction (q, f)
Render the virtual scene or take a photograph of a real world scene illuminated by this directional light source and named $I_{E,q,f}$.

The parameter q is the polar angle, and f is the azimuth. The direction $(0, f)$ is orthogonal to the image plane. The parameters are localized to the coordinate system of the image plane, so transforming the image plane does not affect the BRDF parameterization. The reason for using a directional light source is that the incident light direction is identical at any 3D point. In real life, a directional light source can be approximated by placing a spotlight at a sufficient distance from the scene. The BRDF r of each pixel inside a view can be sampled by the following algorithm,

For each view point E
For each pixel (s,t)

$$r(q, f) = \frac{pixel\ \ value\ \ at\ \ (s,t)\ \ of\ \ I_{Eqf}}{intensity\ \ of\ \ light\ \ source}$$

One assumption is that there is no intervening medium that absorbs, scatters, or emits any radiant energy.

Since the viewing direction of each pixel within one specific view of the image plane is fixed, the BRDF r is simplified to a unidirectional reflectance distribution function (URDF) that depends on the light vector only. Hence, the function r is parameterized by two parameters (q, f) only. From now on, when we refer *BRDF*, we actually mean the URDF as viewed from a certain viewpoint.

Traditionally, the BRDF is sampled only on the upper hemisphere of surface element, since reflectance must be zero if the light source is behind the surface element. However, in our case the reflectance may be nonzero even the light source direction is from the back of the image plane. This occurs because the actual object surface may not align with the image plane (Figure 10). Instead, the whole sphere surrounding the pixel has to be sampled for recording its BRDF. Therefore the range of q should be $[0,\pi]$. Nevertheless, sampling only the upper hemispherical BRDF is usually sufficient, since the viewer seldom moves the light source to the back of objects.

$$\sum_i^n r_i^0 I_i + \sum_i^n r_i^1 I_i + \cdots + \sum_i^n r_i^k I_i$$
$$= \sum_j^k r_0^j I_0 + \sum_j^k r_1^j I_1 + \cdots + \sum_j^k r_n^j I_n$$
$$= r_0 I_0 + r_1 I_1 + \cdots + r_n I_n$$

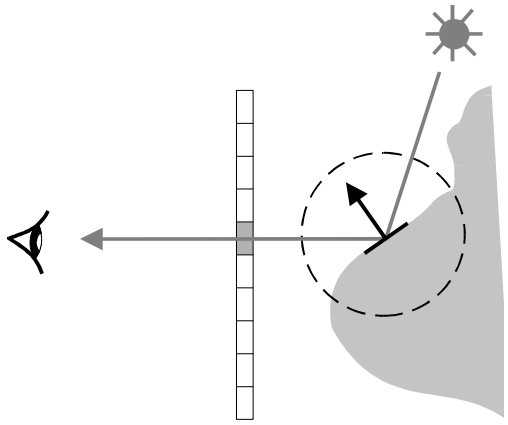


Figure 10: The image plane may not be parallel with the object surface.

3.3 MANIPULATING THE LIGHT SOURCES

Once the BRDFs are sampled and stored, they can be manipulated. The final radiance (or simply value) of each pixel is determined by evaluating equation (1), given the intensity and the direction of the light sources. value at pixel (*s*,*t*) in a view (*E*) = (1) where *n* is the total number of light sources, (q_i, f_i) specify the direction of the *i*-th light source L_i, I_i is the intensity of the *i*-th light source.

$$\sum_i^n r_{E,s,t}(q_i, f_i) I_i \qquad (1)$$

Note that this equation gives a physically correct image. This can be proved as follows. Consider *k* unoccluded objects, visible through the pixel *(s,t)*, viewed from viewpoint *E* and illuminated by *n* light sources. The radiance passing through the pixel window in this view will be

$$r_i = \sum_{j=1}^{k} r_i^j$$

where r_i^j is the reflectance of the *j*-th object illuminated by the light. L_i is the aggregate reflectance we recorded when measuring the BRDF of pixel.

Light Direction: With Equation (1) the light direction can be changed by substituting a different value of (q, f). Figures 14(a) and 14(b) show a teapot illuminated by a light source from the top and the right respectively.

Light Intensity: Another parameter to manipulate is the intensity of the light source. This can be done by changing the value of I_i for the *i*-th light source. Figure 14(c) shows the Beethoven statue illuminated by a blue light from the left.

Multiple Light Sources: We can arbitrarily add any number of light sources. The trade-off is the computational time. Our current prototype can still run at an acceptable interactive speed using up to three directional light sources. In Figure 14(d), the Beethoven statue is illuminated by a blue light from the left and a red light from the right simultaneously.

Type of light sources: Up to now, we have made an implicit assumption that the light source for manipulation is directional. Directional light is very efficient in evaluating equation (1) because all pixels on the same image plane are illuminated by light source from the same direction (q_i, f_i). Despite this, the method is not restricted to directional light. It can also be extended to point source and spotlight. However, it will be more expensive to evaluate equation (1) for other types of light sources, since (q_i, f_i) will need to be recalculated from pixel to pixel.

Since the image plane where the pixels are located is only a window in the 3D space (Figure 9), the intersected surface element that actually reflects the light may be located on any point on the ray *V* in Figure 11. To find the light vector *L* correctly for other types of light sources, the intersection point of the ray and the object surface have to be located first. Note that there is no such problem for directional source, since the light vector is the same for all points in the 3D space. One way to find *L* is to use the depth image. While this can be easily done for rendered images, real world scenes may be more difficult. Using a range scanner may provide a solution. Figures 15(a) and 15(b) show a box on a plane illuminated by a point source and a directional source respectively. Note the difference in the shadow cast by these sources. Other light source types derived from point source can also be used to re-render the image. Figures 15(c) and 15(d) CDROM show an attic scene illuminated by a spotlight and a slide projector respectively. Note how the illumination is correctly accounted for foreground and background objects in both cases.

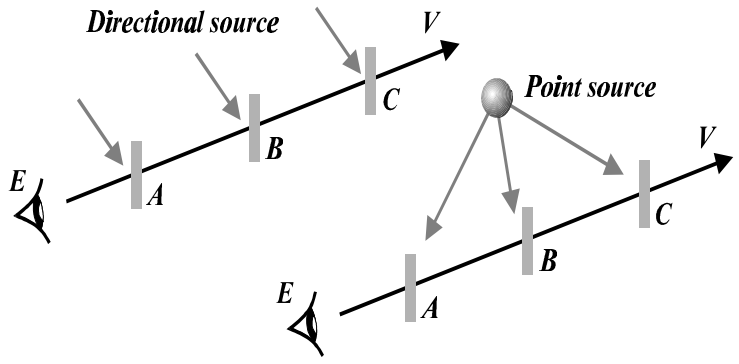


Figure 11: Finding the correct light vector.

4. Conclusions and Future Direction

In this paper we described how two important capabilities, navigation and illumination control, can be achieved in image-based virtual reality applications. We proposed a triangle-based image-warping algorithm, which solves the visibility problem without using depth buffering. By grouping pixels to form triangles, the image warping can be sped up using existing graphics hardware. Moreover, the gap problem of pixel-based approach is also removed at the same time. Both the graph construction and topological sorting have a linear time complexity. Moreover the graph construction and topological sorting are required only when the user changes his/her navigation direction.

We have also proposed and implemented a new image representation to allow the image-based objects to be displayed under varying illumination. The new representation does not restrict the shape or the surface properties of objects in the scene. The objects can be concave and highly specular. Moreover, the illumination is controllable.

In some cases "holes" (unfilled pixels) will still exist even though multiple images are warped and blended together. The sampling scheme (placement of the camera during the capturing phase) requires further investigation. We have also extended the visibility algorithm to panoramic images [7], which are commonly used in commercial virtual reality software. Currently, all of our test data consist of synthetic scenes. We are undertaking the capture of real world scenes with a hand held camera. Compression is another major problem to solve in the future. This is especially important when illumination is included. A few attempts have been made in our previous work [23, 22, 24]. There is still much work to do in using the image-based object as a basic rendering primitive in virtual reality, and our work is only a preliminary step in this direction.

5. Acknowledgement

This work is supported by Hong Kong Research Grants Council CRCs Scheme grant no. CRC4/98.

REFERENCES

[1] Peter N. Belhumeur and David J. Kriegman. "What is the set of images of an object under all possible lighting conditions?" In Proc. IEEE Computer Vision and Pattern Recognition, June 1996.

[2] Brian Cabral, Nelson Max, and Rebecca Springmeyer. "Bi-directional reflection functions from surface bump maps". In Proc. SIGGRAPH '87, Volume 21, pp.273-281, July 1987.

[3] Shenchang Eric Chen. "QuickTime VR - an image-based approach to virtual environment navigation". In Proc. SIGGRAPH'95, pp.29-38, August 1995.

[4] Shenchang Eric Chen and Lance Williams. "View interpolation for image synthesis". In Proc. SIGGRAPH '93, Volume 27, pp.279-288, August 1993.

[5] Michael J. Dehaemer and Michael J. Zyda. "Simplification of objects rendered by polygonal approximations". Computer & Graphics, 15(2):175-184, 1991.

[6] Oliver Faugeras. Three-Dimensional Computer Vision: A Geometric Viewpoint. MIT Press, Cambridge, MA, 1993.

[7] Chi-Wing Fu, Tien-Tsin Wong, and Pheng-Ann Heng. "Computing visibility for triangulated panoramas". In Proc. 10-th Eurographics Workshop on Rendering, pp.169-182, Granada, Spain, June 1999.

[8] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. "The lumigraph". In Proc. SIGGRAPH'96, pp.43-54, August 1996.
[9] James T. Kajiya. "Anisotropic reflection models". In Proc. SIGGRAPH '85, Volume 19, pp.15-21, July 1985.
[10] Donald Knuth. "The Art of Computer Programming; Volume 1: Fundamental Algorithms". Addison-Wesley, 1968.
[11] Stephane Laveau and Olivier Faugeras. "3-D scene representation as a collection of images". In Proc. ICPR'94, pp.689-691, Jerusalem, Israel, October 1994.
Marc Levoy and Pat Hanrahan. "Light field rendering". In Proc. SIGGRAPH'96, pp.31-42, August 1996.
William R. Mark, Leonard McMillan, and Gary Bishop. "Post-rendering 3D warping". In Proceedings of the 1997 Symposium on Interactive 3D Graphics, pp.7-16, April 1997.
[14] Leonard McMillan. "An Image-Based Approach to Three-Dimensional Computer Graphics". Ph.D. thesis, Department of Computer Science, University of North Carolina at Chapel Hill, North Carolina, 1997.
[15] Leonard McMillan and Gary Bishop. "Plenoptic modeling: An image-based rendering system". In Proc. SIGGRAPH '95, pp.39-46, August 1995.
[16] Jeffry S. Nimeroff, Eero Simoncelli, and Julie Dorsey. "Efficient re-rendering of naturally illuminated environments". In 5-th Eurographics Workshop on Rendering, pp.359-373, June 1994,
[17] Lori Scarlatos and Theo Pavlidis. "Hierarchical triangulation using cartographic coherence". CVGIP: Graphical Models and Image Processing, 54(2): 147-161, March 1992.
[18] S. M. Seitz and C. R. Dyer. "View morphing". In Proc. SIGGRAPH '96, pp.21-30, 1996.
[19] J. Shade, S. Gortler, L. He, and R. Szeliski. "Layered depth images". In Proc. SIGGRAPH '98, pp.231-242, July 1998.
[20] Gregory J. Ward. "Measuring and modeling anisotropic reflection". In Proc. SIGGRAPH '92, Volume 26, pp.265-272, July 1992.
[21] Mark Allen Weiss. "Data structures and algorithm analysis". Benjamin/Cummings Pub. Co, 1992.
[22] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or, and Wai-Yin Ng. "Illuminating image-based objects". In Pacific Graphics'97, pp.69-78, Seoul, Korea, October 1997.
[23] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or, and Wai-Yin Ng. "Image-based rendering with controllable illumination. In Eighth Eurographics Workshop on Rendering", pp.13-22, Saint Etienne, France, June 1997.
[24] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or, and Wai-Yin Ng. "Illumination of image-based objects". Journal of Visualization and Computer Animation, 9:113-127, 1998.

BIOGRAPHIES

Tien-Tsin Wong is a visiting assistant professor in the Computer Science Department of Hong Kong University of Science & Technology since August 1998. He graduated from the Chinese University of Hong Kong in 1992 with a B.Sc. degree in computer science. He received his M.Phil. and Ph.D. in computer science from the same university in 1994 and 1998 respectively. His research interests include image-based rendering, photorealistic rendering, medical visualization and natural phenomena modeling.

Contact information:

Dr. Tien-Tsin Wong
Computer Science Dept
Hong Kong University of Science & Technology
Clear Water Bay, Hong Kong.
Phone: +852-23587021
Fax: +852-23581477
Email: mailto:ttwong@acm.org

Chi-Wing Fu, Philip joined the Computer Graphics Lab. at the Indiana University at Bloomington as a Ph.D. student in August 1999. He received his B.Sc and M.Phil in the Chinese University of Hong Kong (CUHK). His research interests include image-based rendering, networked virtual environment, scientific visualization, and medical visualization.

Contact information:

Chi-Wing Fu, Philip.
Computer Graphics Lab
Indiana University at Bloomington, USA.
Email: mailto:cwfu@acm.org

Pheng-Ann Heng is currently an associate professor at the Department of Computer Science and Engineering at the Chinese University of Hong Kong. He received his B.Sc. in computer science from the National University of Singapore, his M.Sc. in computer science, M.A. in mathematics, and his Ph.D. in computer science from Indiana University, U.S.A. He worked as a research scientist at the Institute of Systems Science of the National University of Singapore before he joined the Chinese University of Hong Kong in 1995. He was a visiting scientist at The Johns Hopkins Medical School during 1993 and 1994. He is the Director of the "Virtual Reality, Visualization, and Imaging Research Center" at CUHK and leads several research projects in developing virtual environments for surgical simulation and realtime scientific visualization. His current research interests include virtual reality applications in medicine, interactive scientific visualization, 3D medical imaging, 3D user interface and computer graphics.

Contact information:

Prof. Pheng-Ann Heng
Dept. of Computer Science & Engineering,
The Chinese University of Hong Kong
Shatin, Hong Kong.
Phone: +852-26098424
Fax: +852-26035024
Email: mailto:pheng@cse.cuhk.edu.hk