3 Ray-oriented Buffer for Linear Light Sources

This chapter describes a fast shadowing algorithm for linear light sources that uses a ray-oriented buffer. Space segmentation by the buffer guarantees that if a point is included in a subspace, all light rays toward the point are also contained in the subspace. Each cell of the buffer stores a list of objects that lie within or intersect the subspace allocated to the cell. Therefore, candidate objects, those that may cast shadows onto a point, are determined by referring to the cell where the point is mapped. In addition, whether each candidate object actually casts shadows or not is tested with the bounding-volume of the shadow space to reduce the number of objects subjected to expensive light clipping. The bounding-volumes are also stored in the buffer. For efficiently generating the ray-oriented buffer, we present the cylindrical scan-conversion algorithm. The algorithm pre-converts objects' surfaces to trapezia to decrease the light clipping cost, then connects the trapezia to the buffer cells.

Due to the above improvements, our algorithm achieves over 10 times faster shadow generation compared to the conventional methods. Experimental results confirm that our method can generate realistic images with soft shadows in a few minutes.

3.1 Introduction

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. Hence this chapter presents a fast and accurate shadowing algorithm for linear light sources.

Sampling methods, which convert a linear light source to a series of point lights, are often used. This is because shadowing algorithms for point light sources are so simple that their computation costs are relatively small. However, if the sampling is sparse, digital errors cause serious aliasing artifacts. On the other hand, if it is unnecessarily dense, shadowing cost is excessive. To avoid this sampling problem, analytic shadowing methods are desired.

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.

Nishita *et al.* [Nishta85b] proposed the following algorithm to reduce light clipping cost. Here the case that a polyhedron casts shadows onto a plane is reviewed. First, contours of the polyhedron are projected onto the plane from both end points of the linear light source. As a result, two shadow polygons are generated. Next their convex hull is computed. Finally, light clipping is done merely inside the convex hull, because the whole light source is visible out side of it. This algorithm successfully eliminates redundant intersection tests. However, since each object surface involves generating convex hulls of all other objects, the computing cost remains expensive. In particular, when many complex objects cast shadows onto rugged object surfaces, the cost is extremely high.

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.

Space subdivision methods can also decrease the number of objects subjected to light clipping. For instance, Poulin *et al.*[Poulin90] proposed two methods. One uses a light buffer that is represented as an infinite cylinder oriented along the linear light and

subdivided in arcs along its radius. Each arc saves a list of objects contained within the limit of its angle. Hence a list of candidate objects, which may darken a point, is given by referring to the arc where the point is mapped. Since this computing cost is cheap, the light buffer efficiently reduces the number of objects. However, the reduction is not sufficient. The light buffer is not subdivided along the linear light, so that each cell must cover an infinite wedge shaped subspace. As a result, only a few of the candidate objects actually cast shadows onto the reference point.

The other method scan-converts the light triangle in a 3D grid to determine intersecting voxels, then gathers objects stored in the voxels. This method also reduce the number of objects, however, scan-conversion is very expensive. The light buffer by definition minutely segments 3D space near the light source. By contrast, the 3D grid uniformly subdivides the 3D space, so that its resolution is relatively fine far from the light source. Therefore, some combination of the two methods might give good results, although it was not evaluated.

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.

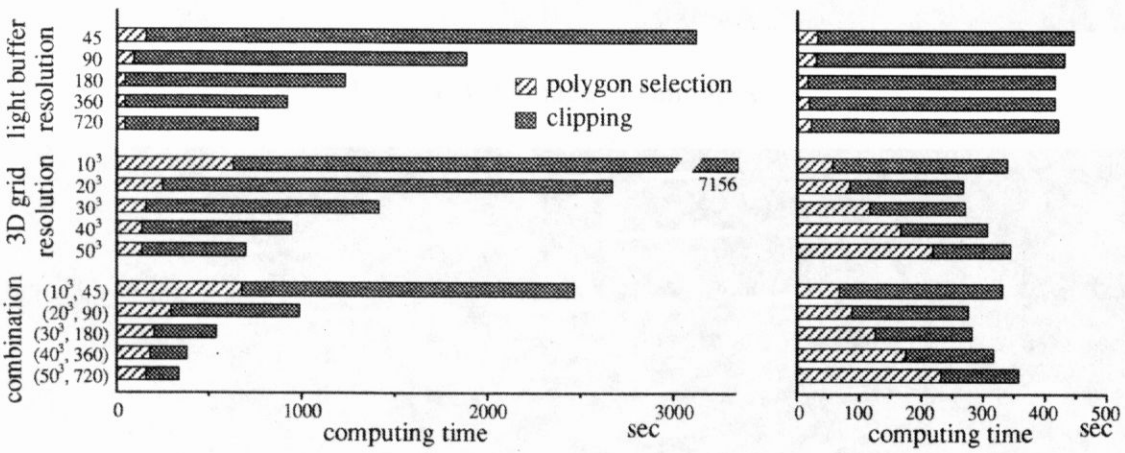
As a matter of course, our shadowing algorithm uses a 2D ray-oriented buffer. The next section estimates the performance of conventional space subdivision methods. Our ray-oriented buffer will then be explained with buffer generation and shadowing algorithms. Finally, performance of our algorithm will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

3.2 Performance of the Conventional Methods

Three subdivision methods; light buffer, 3D grid, and both combination were evaluated. Here, the combination method reduces the number of objects subjected to

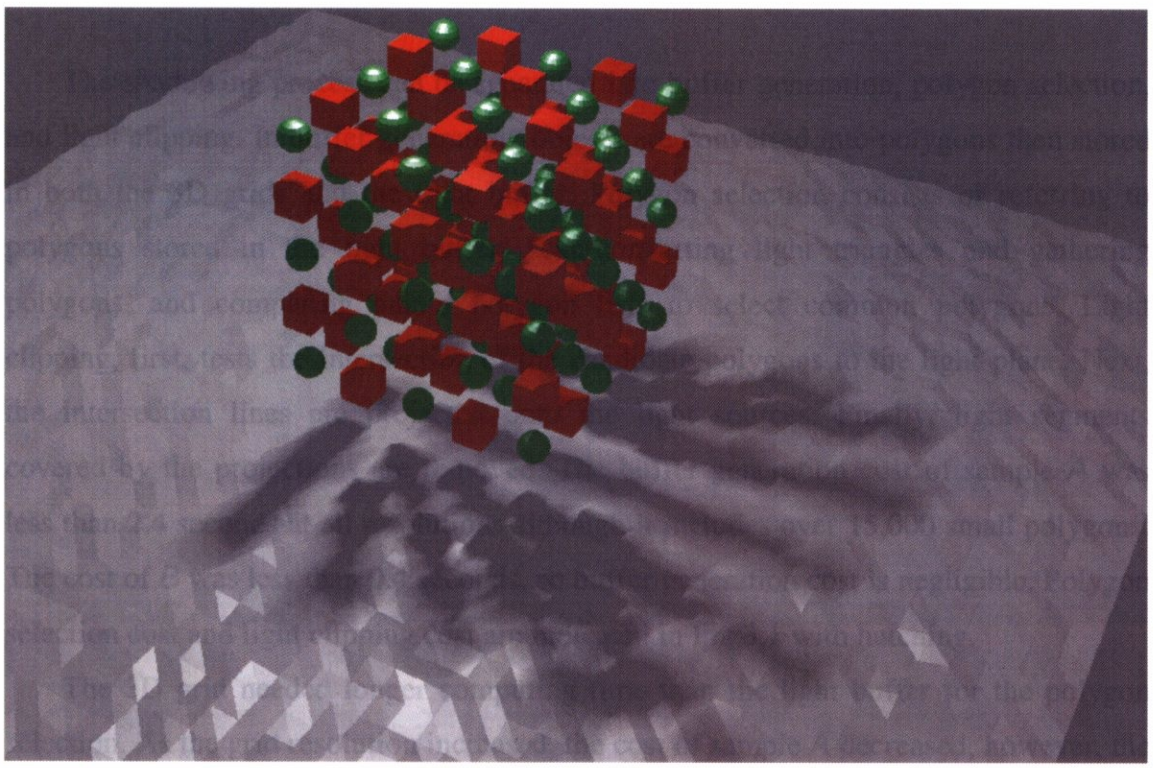
light clipping by selecting common members in two object lists obtained from the light buffer and from the 3D grid. The methods were tested with two samples *A* and *B*. They are shown as Pictures **3.1** and **3.2**, respectively. Sample *A* was chosen because it was expected to be suitable for space subdivision; *B* was selected for its difficulty. Objects in both samples are described as polyhedra.

Their computing costs were measured by varying subdivision resolutions of the 3D grid and the light buffer. Some of the results are presented in Fig.3.1. Sample *A* confirmed the fact that higher resolutions strongly decrease the computing time. Both the light buffer and the 3D grid worked well and their combination was much better. However, the combination method needed 325 seconds even at the highest resolution in Fig.3.1. Shadowing cost of sample *B* was also expensive. According to our expectation, subdivision was not effective.

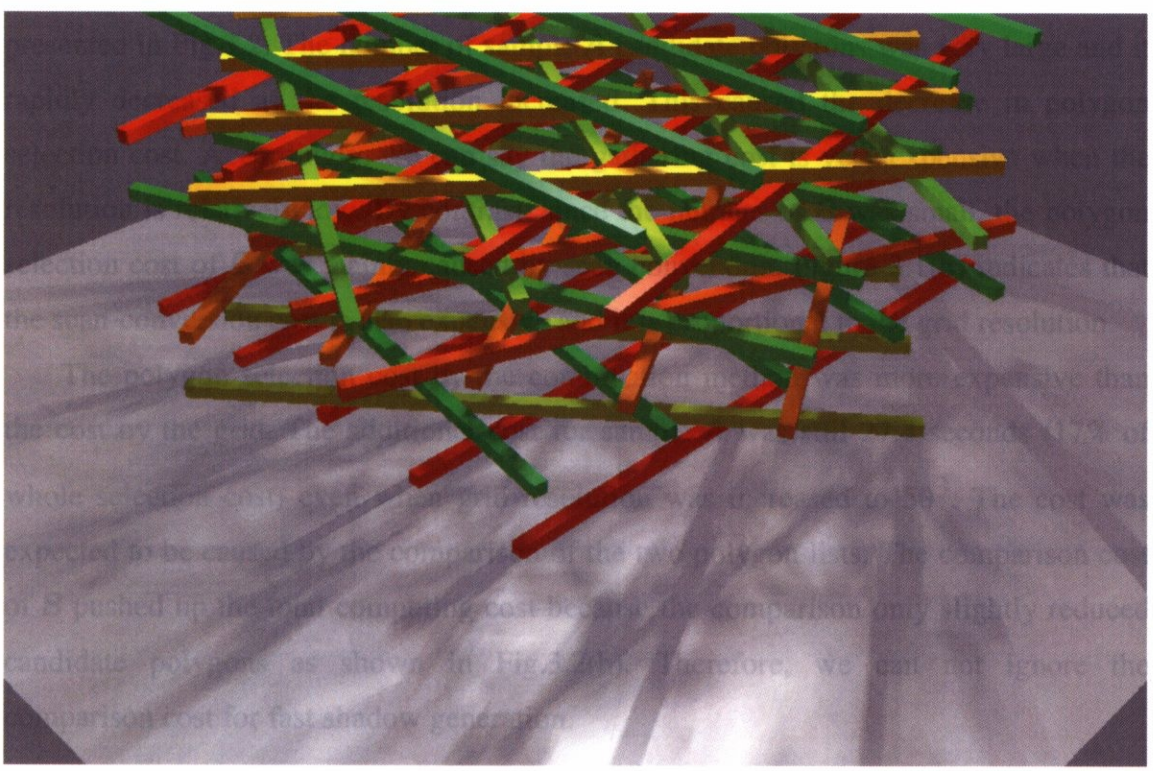


(a) sample A (b) sample B

Fig.3.1 Shadowing cost of the conventional methods.



Picture 3. 1 Image of sample A.

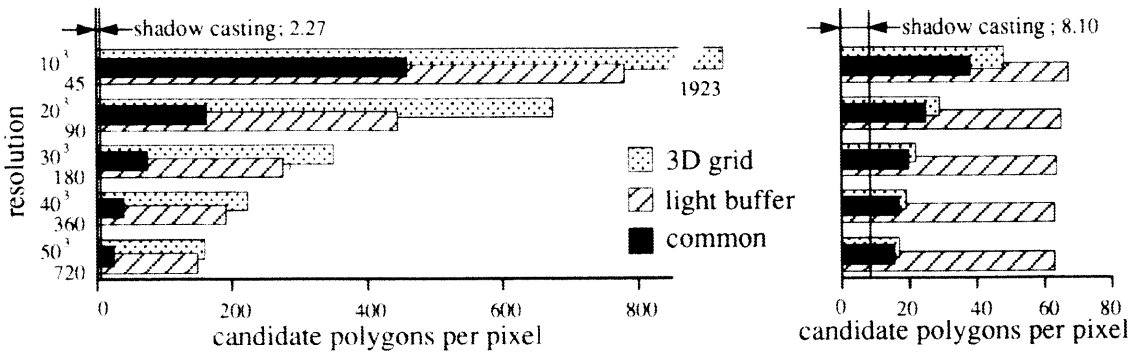


Picture 3. 2 Image of sample B.

The shadowing process can be separated into buffer generation, polygon selection, and light clipping. In buffer generation, objects are converted into polygons then stored in both the 3D grids and the light buffer. Polygon selection consists of referring to polygons stored in the light buffer, scan-converting light triangles and gathering polygons, and comparing paired polygon lists to select common polygons. Light clipping, first, tests the intersection of the candidate polygons to the light plane. Next, the intersection lines are projected onto the light sources. Finally, light segments covered by the projections are removed. The buffer generation cost of sample *A* was less than 2.4 seconds at all resolutions although it includes over 15,000 small polygons. The cost of *B* was less than 0.4 seconds, so buffer generation cost is negligible. Polygon selection cost and light clipping cost are indicated in Fig.3.1 with hatching.

The 3D grid needed longer computing time than the light buffer for the polygon selection. As the grid resolution increased, the cost of sample *A* decreased, however, the cost of *B* increased. To assess this result, we counted the number of candidate polygons per pixel, because polygon gathering cost is proportional to the number. The results are presented in Fig.3.2. The number of sample *A* candidate polygons was very large and it rapidly decreased as the resolution increased. This caused the decrease in polygon selection cost. Accordingly, we can say the gathering cost is very expensive when the resolution is relatively low. By contrast, since the number of *B* was small, the polygon selection cost of *B* was mainly caused by scan-conversion. Figure 3.1(b) indicates that the scan-conversion cost is also expensive and it is proportional to the grid resolution.

The polygon selection cost by the combination method was more expensive than the cost by the grid. The additional cost for sample *A* was still 27.4 seconds (17% of whole selection cost) even when grid resolution was increased to 50^3. The cost was expected to be caused by the comparison of the two polygon lists. The comparison cost of *B* pushed up the total computing cost because the comparison only slightly reduced candidate polygons as shown in Fig.3.2(b). Therefore, we can not ignore the comparison cost for fast shadow generation.



(a) sample A (b) sample B

Fig.3.2 Average candidate polygons selected by the conventional methods.

The light clipping cost is a function of the number of candidate polygons. As shown in Fig.3.2, fine space subdivision reduced the number, so the clipping cost drops in Fig.3.1. However, the light clipping cost is still expensive even if the highest resolution is chosen. One of the causes is insufficient elimination of candidate polygons. Sample *A* retains 26.1 candidate polygons per pixel at best, while only 2.27 of them actually intersected the light triangle. This means that more than 10 times as many polygons were tested as was necessary. Half the candidate polygons of *B* were also unnecessarily evaluated. Another cause is that the light clipping operation itself requires much computation. The cost for sample *B* was estimated to be 45.1 μsec/polygon. The cost was primarily spent on determining the intersections of polygons and the light plane.

3.3 Ray-oriented Buffer for Fast Shadowing

The experimental results in Section **3** indicate that fast shadowing needs

(1) space subdivision to eliminate expensive 3D scan-conversion and polygon comparison,

(2) accurate candidate polygon selection to avoid unnecessary computation, and

(3) fast intersection calculation between the candidate polygons and the light plane.

Each of these issues will be resolved as follows.

3.3.1 Ray-oriented Segmentation

The cylindrical coordinate system is generally described as (r,θ,z), where z is the position on the **Z** axis, θ is the rotational angle around the **Z** axis, and r is the distance from the **Z** axis. To simplify descriptions, the world coordinate system is converted to the cylindrical coordinate system with the mapping function which projects the linear light source to the line segment from (0,0,-1) to (0,0,+1).

Light rays that projected from a linear light source toward an illuminated point generally intersect many voxels of the 3D grid. Thus the light triangle must be scan-converted in the grid to determine all intersecting voxels and then to collect all candidate polygons. If all of the light rays radiated to a point are completely kept inside a single subspace embracing the point, the candidate polygons are directly obtained by referring to the subspace. Thus the scan-conversion is not needed. To satisfy this inclusion condition, we segment 3D space with sectors and bands as follows.

As shown in Fig.3.3, the 3D space is split by planes including the **Z** axis (*i.e.* the light source) and separated into sectors. The i-th sector is the arc-like subspace defined as $2\pi(i-1)/n \le \theta < 2\pi i/n$, here n is the number of sectors and $i \in [1, n]$. Since each sector is designed as a wedge originating from the light source, the light triangle for a point is contained in the sector including the point. Segmentation by the sector is equivalent to Poulin's light buffer.

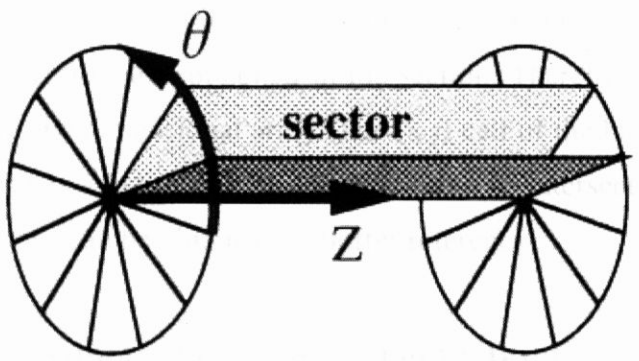


Fig.3.3 Sector.

The band is a subspace bounded by a pair of cones rotating around the **Z** axis. The band is defined with two angles; ϕ and ψ shown in Fig.3.4. Here, ϕ is the angle of the

semiline terminating at (0,0,+1) from the +**Z** direction and ψ is the angle of the semiline terminating at (0,0,-1) from the -**Z** direction. When the band resolution is m, the j-th band as $j \in [1, m]$ satisfies $\pi(j-1)/m \leq \phi$ and $\pi(m-j)/m \leq \psi$. The bands segment 3D space, however, they overlap each other. As shown in Fig.3.5, if a point is located in a band, all light rays toward the point are also included in the band.

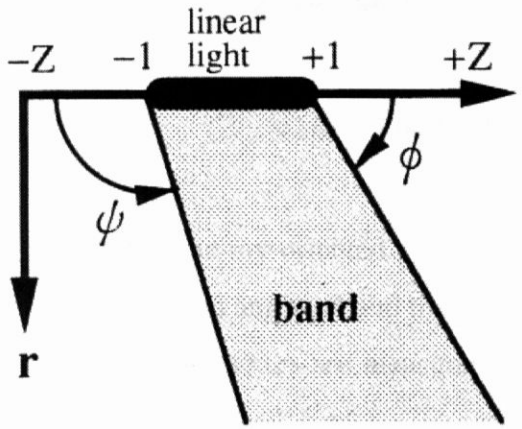


Fig.3.4 Band.

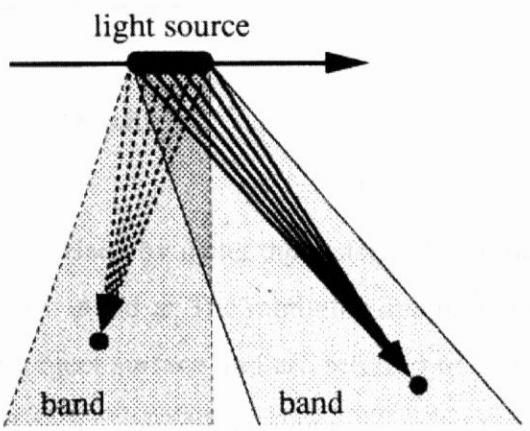


Fig.3.5 Light rays toward a point.

Sectors and bands together segment 3D space into many subspaces named sections. The (i, j) section is defined as intersection of the i-th sector and the j-th band. Since both sectors and bands satisfy the inclusion condition, this segmentation guarantees that rays toward any point in a section are contained in the section. Therefore, our ray-oriented buffer is designed as a two dimensional array. The (i, j) cell of the buffer is assigned to the (i, j) section and saves a list of objects that lie within or intersect the section. As a result, candidate polygons can be obtained by buffer reference.

3.3.2 Candidate Reduction by Using Shadow Bounding-Volumes

As shown in Fig.3.6, the ray-oriented buffer is sparser than the 3D grid at points far from the light source. This, unfortunately, gives more objects than the conventional method, so we introduce 5 formulas to reduce object number.

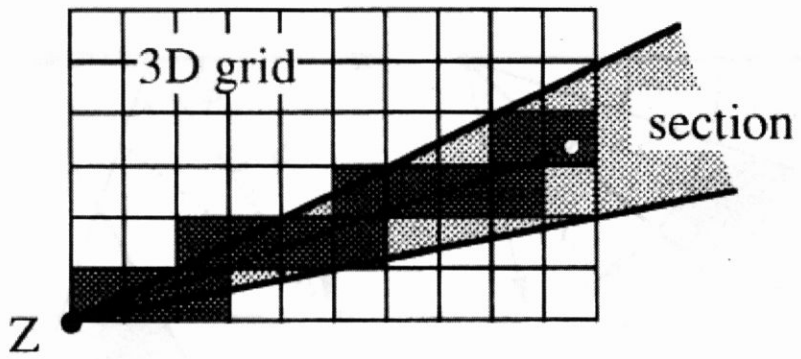


Fig.3. 6 Covered 3D space.

A point in the ray-oriented buffer is characterized by using the distance from the **Z** axis: r, the rotation angle: θ, and the band angles: ϕ and ψ. The minimum and maximum values of the parameters are measured on an object surface and are referred to as r_{min}, θ_{min}, θ_{max}, ϕ_{max}, and ψ_{max}. These parameters define a bounding-volume of shadow space caused by the object, which is illustrated in Fig.3.7. In general, the values at point P are described as r_P, ϕ_P, ψ_P, and θ_P. Whenever the object casts shadows (including penumbras) onto point **P**, the following 5 formulas are satisfied;

(1) $r_P \geq r_{min}$,

(2) $\phi_P \leq \phi_{max}$,

(3) $\psi_P \leq \psi_{max}$,

(4) $\theta_P \geq \theta_{min}$, and

(5) $\theta_P \leq \theta_{max}$.

Therefore, if an object misses even one formula, it can be eliminated because it never casts shadows onto the point. The formulas limit object number much more severely than the 3D grid.

The values r_{max}, ϕ_{max}, ψ_{max}, θ_{max}, and θ_{min} are preprocessed and saved in our buffer. Thus, an object requires the storage of 7 words per section for memorizing the 5 values, object ID, and list pointer. Storage volume can be reduced with hierarchial descriptions. On the other hand, the conventional method uses at least 4 words; 2 IDs and 2 pointers for the light buffer and the 3D grid.

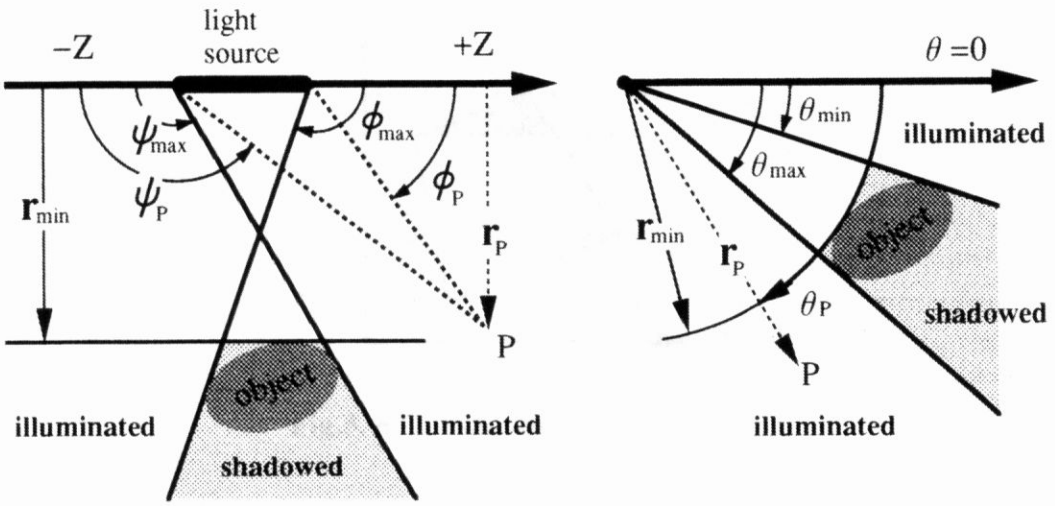


Fig.3.7 Limitation of illuminated space.

3.3.3 Preconversion into Trapezia for Fast Intersection Calculation

Every polygon is divided by planes before it is stored in the ray-oriented buffer. Each plane commonly includes the linear light source and passes through a polygon vertex. As shown in Fig.3.8, a polygon is converted into a set of trapezia and/or triangles. This conversion reduces the computing cost of light clipping. The reason is as follows. If the light plane intersects a trapezium, the rotation angle θ of the plane must be between the bottom θ and top θ of the trapezium. Thus whether the plane intersects the trapezium or not is easily judged by comparing the θ values. Moreover, the intersection line between them is quickly calculated because intersecting polygon boundaries have been uniquely determined; the trapezium does not need expensive computation to determine the boundaries. In consequence, our buffer stores the resulting triangles and trapezia instead of the original polygons. This technique can not handle curved objects directly, however, since objects are normally tesselated in image generation, this dividing technique is practical and worthwhile.

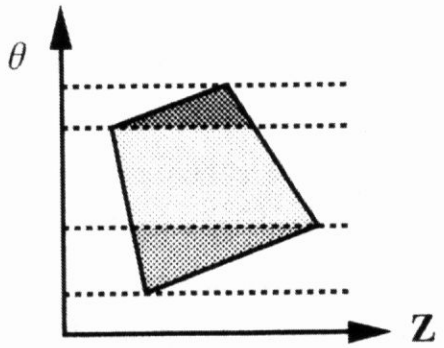


Fig.3. 8 Division by θ of vertices.

3.3.4 Cylindrical Scan-Conversion for Efficient Buffer Generation

The ray-oriented buffer is the key to realize fast shadow generation. In this section, we present the cylindrical scan-conversion algorithm which efficiently constructs the buffer. Our algorithm uses the cylindrical coordinate system, however, it is based on the scanline algorithm. Accordingly, our algorithm inherits the strong points of the scanline algorithm in terms of accuracy and efficiency. The proposed cylindrical scan-conversion algorithm, described below, is applied to all polygons in the scene.

- If the front side of a polygon is invisible from both light ends, then exit.
- Map polygon vertices into the cylindrical coordinate system.
- If the converted polygon intersects the θ=0 plane, divide the polygon by the plane.
- Save parameters of the polygon edges in the ray-oriented buffer.
- Create the vertex list and sort it in increasing order of θ. The list is described as $\{V_i;\ i=1,n\}$, here n is the number of vertices. The θ value of the i-th vertex is θ_i.
- Empty the polygon edge list **EL.**
- For all i in $[1, n-1]$, {
  - If a polygon edge connecting at V_i is a member of **EL**, remove it, otherwise add it.
  - Pair the edges in **EL** in increasing order of their **Z** values over $[\theta_i, \theta_{i+1}]$. Each edge pair bounds a triangle or a trapezium.
  - For each edge pair, {
    - Store θ_i and θ_{i+1} as the θ_{min} and θ_{max} of the edge pair, respectively.
    - Calculate intersections of sectors to the trapezium (or triangle) bounded by the edge pair. The intersections are called spans.
    - For each span, {
      - Calculate its ϕ_{max} and ψ_{max} and store them in the buffer.
      - Determine sections intersecting to the span.
      - For each section, {
        - Add the span in the ray-oriented buffer with its r_{min} in the section.

      }

    }

  }

}

Fig.3. 9 Outline of Cylindrical Scan-Conversion Algorithm.

After all objects are scan-converted, spans stored in each section are sorted in increasing order of their r_{min}. This sorted span list ensures that the shadow generation for all remaining spans is skipped whenever the formula $r_{min} \geq r_P$ is satisfied. Moreover, this arrangement avoids unnecessary light clipping because object near the light source, which possibly hide the light source, are firstly tested in the shadow generation.

3.3.5 Shadowing Algorithm with Ray-oriented Buffer

When a point in the 3D space is given, the light segments illuminating the point are determined by the following algorithm.

- Map the point onto the ray-oriented buffer.
  Its position is described as (r_P, θ_P, z_p).
- Calculate ϕ_P and ψ_P.
- Secure the span list from the section including the point.
- While the span list is not empty, {
  - Get a span from the list.
  - If $r_{min} \geq r_P$, then exit this loop.
  - If $\theta_{min} \leq \theta_P \leq \theta_{max}$ and $\phi_{max} \geq \phi_P$ and $\psi_{max} \geq \psi_P$, {
    - Calculate intersection of the span and the light plane.
    - Project the intersection onto the linear light.
    - Remove segments covered by the projection.
    - If the light source is completely hidden, then stop. /* shadowed */

    }

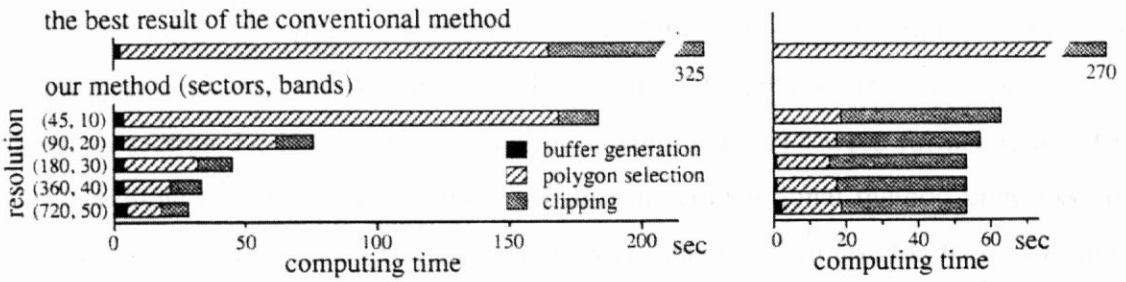
  }
- Compute intensity. /* illuminated or in penumbras */

Fig.3.10 Outline of Shadowing Algorithm with Ray-oriented Buffer.

3.4 Experimental Results

3.4.1 Performance Comparison

Performance of the cylindrical scan-conversion algorithm and the shadow generation algorithm were examined using examples *A* and *B*, which were introduced in Section 3.2. The results are presented in Fig.3.11. They were compared to the results of the conventional methods shown in Fig.3.1. The results confirm that the ray-oriented buffer can strongly reduce shadow generation cost. In sample *A*, the shortest computing time of our method (that is 28.2 seconds) is 8.7% of the best result by the conventional method (325 seconds). In sample *B*, the time by our method (53.5 seconds) is 80.4% less.



(a) sample A **(b)** sample B

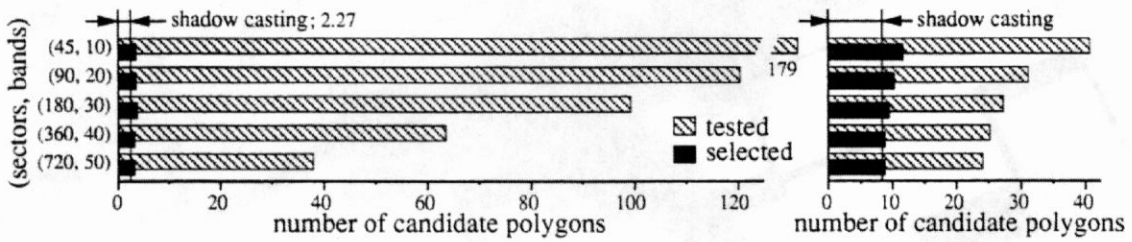
Fig.3.11 Shadowing cost of our method.

It is anticipated that the ray-oriented buffer would need longer generation times because it is more complex then the previous buffers. The buffer generation cost was proportional to both sector and band resolution. However, even if the ray-oriented buffer is most precisely resolved, *i.e.* 720 sectors×50 bands, the cost is 5.3 seconds for sample *A*. This is only 19% of the total cost of *A*. In sample *B*, the cost is negligibly small because it is 4% of the total cost.

In candidate polygon selection, our shadowing algorithm was much faster than the combination method of the previous light buffer and the 3D grid. The cost for picking up candidates from the 720 sectors×50 bands buffer was 12.7 seconds for sample *A* and 16.3 seconds for *B*. These values are 7.8% and 7.0% of the costs by the combination method with the comparable resolution, respectively.

The selection cost is logically in proportion to the number of polygons tested. Figure 3.12 presents averages of the numbers. Polygons satisfying the 5 formulas are selected as candidates. The selected numbers are also indicated in Fig.3.12. In general, the number of polygons tested is larger than the number of candidate polygons by the combination method in Section **3.2** at comparable resolution, however, the formulas strongly reduced the number. Actually, Fig.3.12 shows that almost all false candidate polygons were eliminated by the formulas. The remaining false candidates increase light clipping cost by only 26% and 6%. On the other hand, the combination method included 1050% and 92% false candidates, respectively. Therefore, our candidate selection algorithm is quite reliable. In addition, our polygon selection cost is much cheaper than that of the conventional method. Therefore, our algorithm is extremely efficient.

On average, 15.7 μsec was required for a trapezium to clip a light source. This almost one third the time taken by the conventional method (45.1 μsec/polygon as shown in section 3). However, the clipping costs in Fig.3.11 were further reduced to 1/10 in sample *A* and 1/4 in *B*. This is because the conventional method unnecessarily clipped numerous polygons. The number of polygons was 9 times (for *A*) or 1.8 times (for *B*) more than ours.



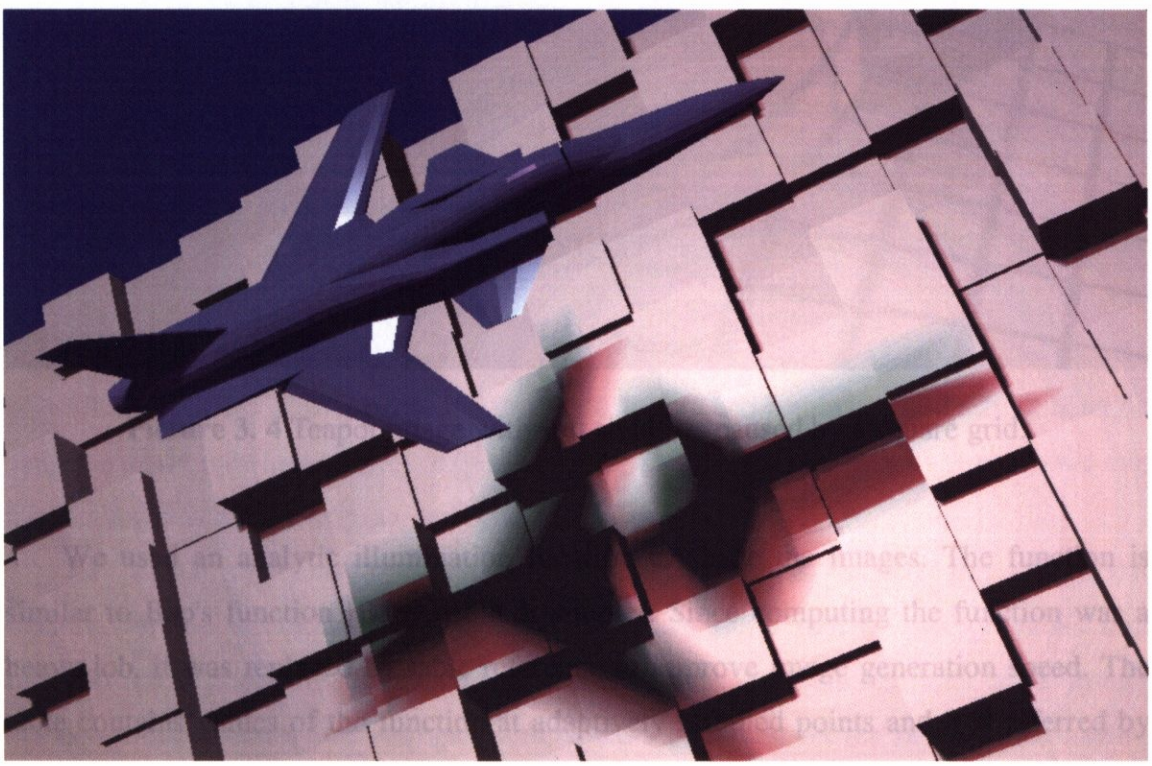
(a) sample A **(b)** sample B

Fig.3. 12 Average of tested and selected polygons by our method.

3.4.2 Image Generation

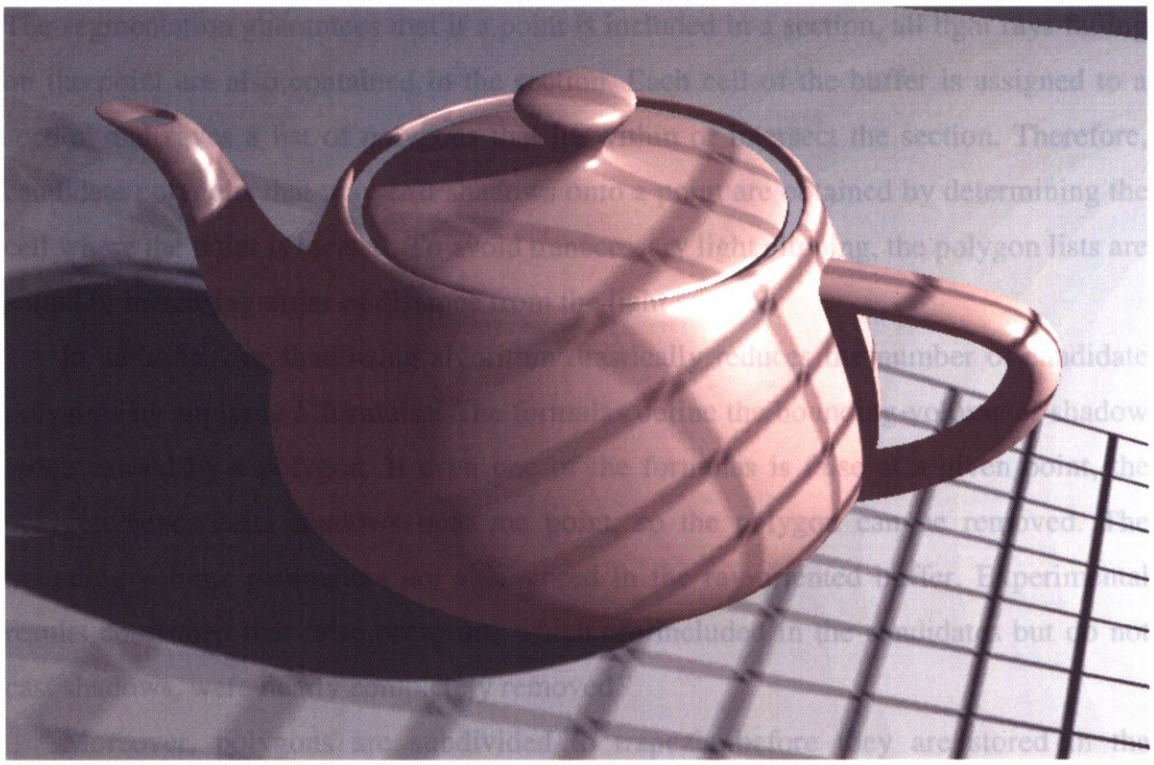
Whole image generation times with our shadowing algorithm were measured. The time for Picture 3.1 was 83.9 seconds on a Sun work station, when the image size was 640·480 pixels and the buffer resolution was 720 sectors·50 bands. The time for Picture 3.2 was 107.3 seconds under the same conditions. These times include data loading and image saving costs. Shadowing costs were actually 28.2 and 53.5 seconds, respectively.

Picture 3.3 is an example of linear light illumination. The airplane was illuminated by 2 linear light sources; green and magenta. The light sources were parallel to the ground plane and crossed each other at right angles. The airplane dropped soft shadows onto square blocks that were randomly pulled up or pushed down. Each block also cast shadows onto the others. The image would seem to need long computation time, however, it was generated in 114.3 seconds with the conditions of 640·480pixels and 720 sectors·50 bands.



Picture 3. 3 Synthesized airplane image with 2 linear light sources; green and magenda.

Picture 3.4 shows shadows cast by a square grid. The shadows were projected onto the teapot and the base plane. The teapot was tessellated with over 7,000 polygons and displayed with Blinn's smooth-shading algorithm [Blinn76]. The teapot also cast shadows onto itself and the base plane. These objects were illuminated by a linear light source rotated at 45 degrees to the grid. The image needed 113.7 seconds with the same conditions. Our algorithm can successfully generate shadows dropped onto curved surfaces in reasonable computing time.



Picture 3. 4 Teapot image with soft shadows caused by a square grid.

We used an analytic illumination function to shade the images. The function is similar to Bao's function [Bao93] but is simpler. Since computing the function was a heavy job, it was replaced by table reference to improve image generation speed. The table contains values of the function at adaptively sampled points and it is referred by using the binary search algorithm [Tanaka93]. This adaptive sampling promises sufficient accuracy in image generation. The analytic illumination function and the acceleration technique with the table will be presented in the near future.

3.5 Conclusion

In this chapter, a fast shadowing algorithm for linear light sources was proposed. This algorithm accomplishes the following improvements.

To reduce the computing cost of candidate selection, our algorithm employs the

ray-oriented buffer. The buffer segments 3D space to many sections with 2 parameters. 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. Therefore, candidate polygons that may cast shadows onto a point are obtained by determining the cell where the point is located. To avoid unnecessary light clipping, the polygon lists are sorted in increasing order of distance from the light source.

In addition, our shadowing algorithm drastically reduces the number of candidate polygons by applying 5 formulas. The formulas define the bounding-volume of shadow space caused by a polygon. If even one of the formulas is false at a given point, the polygon never casts shadows onto the point, so the polygon can be removed. The bounding-volume parameters are also stored in the ray-oriented buffer. Experimental results confirmed that false polygons, which are included in the candidates but do not cast shadows, were nearly completely removed.

Moreover, polygons are subdivided to trapezia before they are stored in the ray-oriented buffer. This is because intersections between trapezia and the light triangle can be quickly calculated. In the experiment, light clipping cost for a trapezium was one third of the cost for the original polygon.

To reduce the cost for generating the ray-oriented buffer, the cylindrical scan-conversion algorithm was introduced. The algorithm first divides polygons into trapezia, then computes intersections of the trapezia to the sections, and finally calculates the bounding-volume parameters and stores them in the ray-oriented buffer. The cylindrical scan-conversion processed 10,000 polygons in about 5 seconds. The cost was sufficiently small compared to the whole image generation cost.

Due to the above improvements, the proposed shadowing algorithm was over 10 times faster than the conventional algorithm. Our shadowing algorithm successfully generated pictures with realistic half shadows. Since, their generation took only a few minutes, the generation speed is acceptable.

The proposed algorithms were designed for polyhedral objects, however our ray-oriented buffer has the ability to handle curved objects. In the case of curved objects, the buffer description of sectors and bands will be retained. The cylindrical scan-conversion can be adopted with a small modification if bounding boxes of curved

objects are given. However, a fast clipping algorithm is a remaining problem. We think Bézier clipping [Nishita90] satisfies our demand but it must be confirmed. In addition, we are planing to adaptively divide the ray-oriented buffer to reduce the memory storage requirement.