4 Extended Ray-oriented Buffer for Area Light Sources

This chapter describes a fast analytic algorithm that generates exact highlights and soft shadows from area light sources. In order to realize fast shadowing, we propose the ray-oriented buffer which segments 3D space by following light rays from polygonal sources. Each cell of the buffer stores objects that intersect a related subspace. Candidate objects which may cast shadows onto a point are selected by referring to the buffer. The candidates are then tested with their shadow bounding volumes to suppress objects that never occlude light sources.

In addition, we propose the cross scanline clipping algorithm. It quickly determines the exact regions of uncovered area light sources with simple silhouette generation. Both diffuse and specular reflections are computed by integrating light rays from the uncovered sources. Experimental results confirm the high performance of the proposed method.

4.1 Introduction

To realize advanced local illumination models for area light sources is one of the most important tasks in the field of photo-realistic rendering. Area 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.

4.1.1 Previous Work

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.

Analytic approaches can completely suppress the aliasing artifacts since they determine unoccluded portions of each area light source exactly. This exact segmentation, termed "*light clipping*", is performed as follows. First, a light volume is defined by all vertices of the polygonal light source and the point at which illumination is to be calculated. Each object is then tested to determine whether it intersects the light volume or not. Finally, occluded light regions are removed if intersection does occur. This light clipping, which is expensive, is needed for all objects in a scene at every pixel.

The methods to reduce the total amount of light clipping processes can be classified into the following three approaches. The first is the penumbra preprocessing approach. 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.

The second approach computes the visibility prior to rendering by using, for example, BSP trees [Chin89] or discontinuity meshing [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. The back-projection techniques [Drettakis94], [Stewart94] are complicated and still take a long time to generate discontinuity meshes.

The third approach is space subdivision. 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.

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.

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.

4.1.2 Proposed Method

Based on the above analysis, we present a fast rendering system by integrating the following techniques:

- extended ray-oriented buffer technique,
- cross scanline clipping algorithm, and
- a rendering equation with an analytic integration of both diffuse and specular intensities.

The 3D space is subdivided by following light rays radiating from polygonal sources. Objects intersecting each subspace are listed in the related cell of a ray-oriented buffer. Thus, objects that possibly cast shadows onto an observed point can be selected by referring the buffer. However, since many false objects still remain, further improvement is needed to precisely suppress them. Therefore, bounding volumes of shadow spaces are introduced. The volumes are also stored in the buffer. This idea was originally proposed for linear light sources as already mentioned in the previous chapter.

In this chapter, we extend them to suit area light sources. In addition, the CCSC (Coupled-Cylindrical Scan Conversion) algorithm is presented for fast buffer generation.

The efficiency of light clipping itself is critical. As Weiller and Atherton [Weiller77] pointed out, uncovered light segments can be obtained by using hidden surface removal algorithms. There has been some progress in this essential algorithm. Most proposals utilize hardware z-buffers to accelerate hidden surface removal process [Baum90], however, due to the need to avoid aliasing artifacts, exact uncovered light pieces must be computed. Thus, we apply the cross scanline algorithm (details are described in Appendix **A**) which accurately determines visible pieces of polygons. The algorithm named cross scanline clipping is also presented in this chapter.

We have already proposed an analytic rendering equation for area light sources (details are described in Appendix **B**), however, shadowing was not supported. The equation integrates both diffuse and specular reflections with Lambertian distributive polygonal light sources, thus we apply the equation to the uncovered light sources.

In the following sections, details of each component are described.

4.2 Ray-oriented Buffer

4.2.1 Ray-oriented Space Subdivision

The surface normal of the area light source is chosen as the **Z** axis. Thus, light rays are projected toward the +**Z** half-space. We first segment the 3D space to many x-bands. As shown in Fig.4.1, an x-band is a subspace bounded by a pair of radial planes. One radiation center $\mathbf{R}^+$ is allocated to the maximum x value of the light source and the other center $\mathbf{R}^-$ is the minimum. The segmentation planes are perpendicular to the **X**-**Z** plane and defined by angles ϕ^+ and ϕ^- measured in the -**X** direction as shown in Fig.4.1.

When the band resolution is m, the i-th band as i in [1, m] satisfies $\pi(i-1)/m \leqq \phi^-$ and $\phi^+ < \pi\, i/m$. Therefore, the x-bands cover the +**Z** half-space although they overlap each other. This band segmentation guarantees that if a point is located in an x-band, all light rays toward the point are also included in the band.

Then y-bands segment the 3D space as do the x-bands, however, segmentation

planes are perpendicular to the **Y-Z** plane. The radiation centers, $\mathbf{S}^+$ and $\mathbf{S}^-$, are allocated to the maximum and the minimum y of the light source, respectively. A y-band is defined by angles γ^+ at $\mathbf{S}^+$ and γ^- at $\mathbf{S}^-$ measured in the **-Y** direction.

The intersection of the i-th x-band and the j-th y-band defines the (i, j) section. Since both x- and y-bands satisfy the ray inclusion condition, light rays toward any point in a section are also contained in the section. A list of objects that lie within or intersect the (i, j) section is stored in the (i, j) cell of our ray-oriented buffer. From the ray inclusion condition, objects casting shadows on a point must intersect the section containing the point.

Thus, candidate objects which may occlude the light source can be obtained by simple buffer reference.

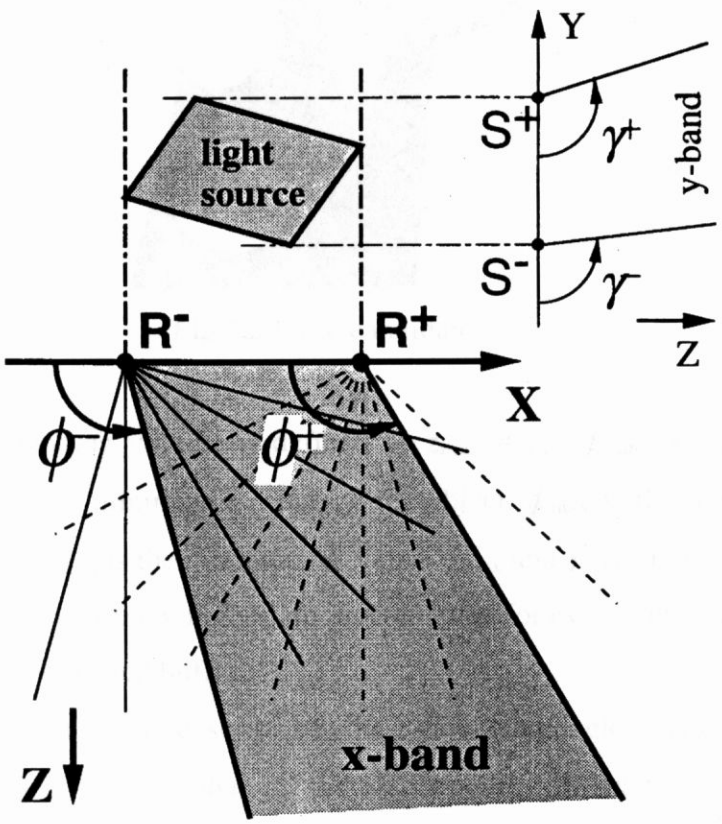


Fig.4.1 Ray-oriented space subdivision.

4.2.2 Shadow Bounding Volume

Some candidate objects actually do not cast shadows onto an observed point. For example, objects **A** and **B** in Fig.4.2 make no shadows on point **P**, although they intersect the band. To reduce these false candidates, shadow bounding volumes are introduced.

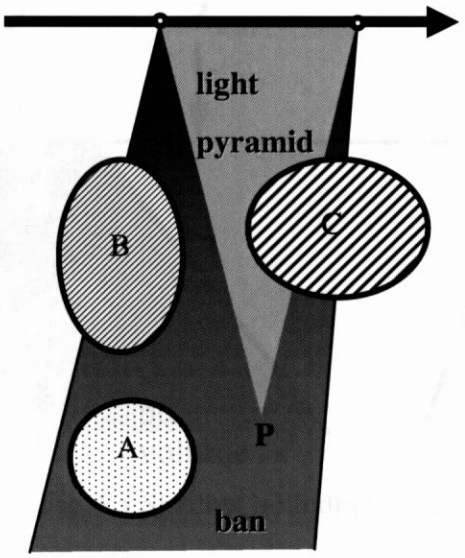


Fig.4.2 False candidates.

The shadow bounding volume is defined by z_{min}, ϕ_{max}, ϕ^{+}_{min}, γ^{-}_{max}, and γ^{+}_{min} in Fig.4.3. Here z_{min} is the minimum z value of the object. ϕ_{max} is the maximum of the x-band angle ϕ, and ϕ^{+}_{min} is the minimum ϕ^{+} value. γ^{-}_{max} and γ^{+}_{min} are the angles for the y-band. The shadow bounding volumes are precomputed for every object at each section, then stored in the ray-oriented buffer.

The observed point is also described by its z value and angles; z_p, ϕ_p, ϕ^{+}_{p}, γ^{-}_{p}, and γ^{+}_{p}. If z_{min} of an object is larger than z_p, the object never cast shadows onto the point. Thus the object **A** in Fig.4.2 is removed. The objects satisfying the **z** condition are then tested with the angles. The angle conditions are: $\phi_{max} \geqq \phi_p$, $\phi^{+}_{min} \leqq \phi^{+}_{p}$, $\gamma^{-}_{max} \geqq \gamma^{-}_{p}$, and $\gamma^{+}_{min} \leqq \gamma^{+}_{p}$. The object **B** in Fig.4.2 is removed by ϕ comparison. The shadow volume test limits object number much more severely than the 3D grid.

The cost of the volume test is sufficiently low because only five comparisons are

needed. In addition, the test is halted when the next object's z_{min} is larger than z_p. This is because the ray-oriented buffer saves objects sorted in increasing order of their z_{min} value.

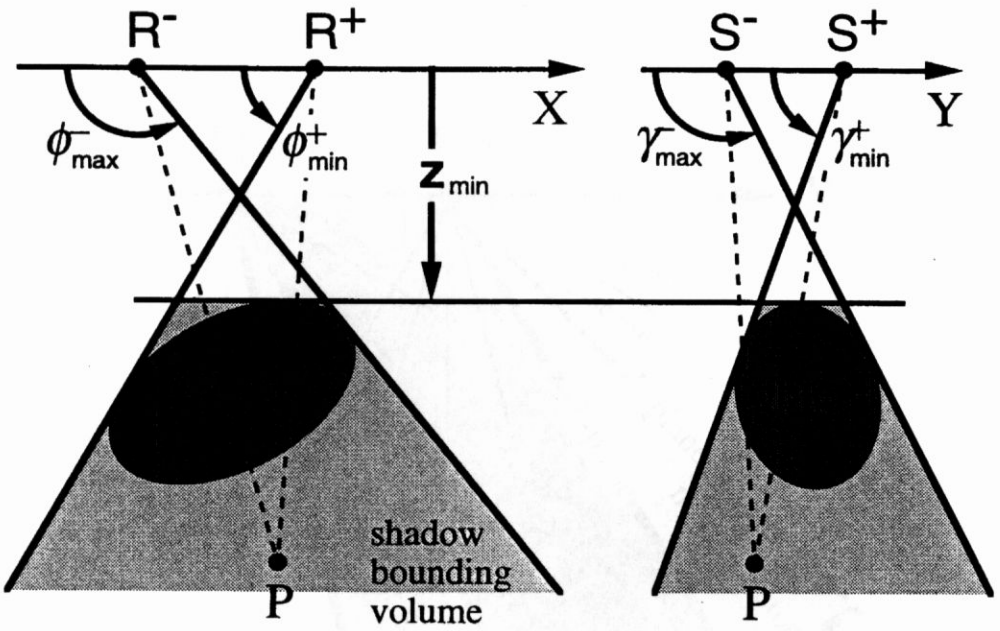


Fig.4.3 Shadow bounding volume.

4.2.3 Coupled Cylindrical Scan Conversion Algorithm

The CCSC (Coupled-Cylindrical Scan-Conversion) algorithm is presented for efficient buffer generation. This algorithm assumes polyhedral objects, thus curved surfaces must be tessellated before conversion. A polygon is stored in the ray-oriented buffer with 2 level subdivision: ***x***-bands and ***y***-bands.

In the CCSC for ***x***-bands, F-planes are radiated from $\mathbf{R_p}$ as shown in Fig.4.4 and **B**-planes are from Rm. These radiation centers are allocated at the maximum and minimum ***x*** values of the light source. The first plane radiates to **-X** direction and the last to **+X**. The planes are perpendicular to the **X-Z** plane. The **F**- and **B**-planes segment the **+Z** space to form many **F**- and **B**-sectors, respectively. Since an ***x***-band is bounded by a pair of **F**- and **B**-planes, it is efficiently updated by adding the next **F**-sector and removing the last **B**-sector as shown in Fig.4.4.

Polygon edges intersecting an **F**-plane and a **B**-plane are contained in the **F**-list and the **B**-list, respectively. Edges that lie within or intersect an ***x***-band are indicated by the

I-list. These three lists access the common edge data. This is important for efficient edge clipping by an ***x***-band or a ***y***-band. To revise the lists in the same manner as the scanline algorithm, each edge of a polygon is saved in the **F**- and **B**-sectors to which its end points are projected.

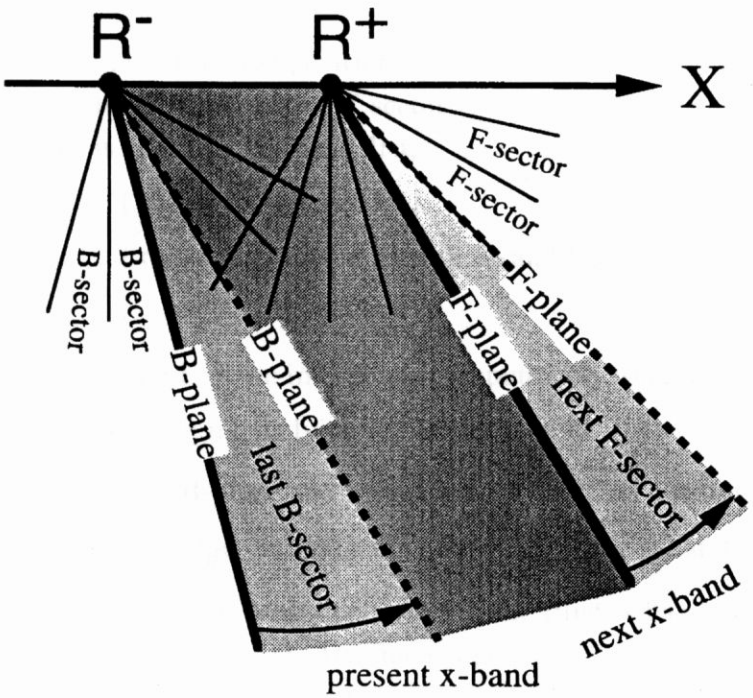


Fig.4.4 Coupled Cylindrical Scan Conversion

The CCSC algorithm described below is applied to each polygon in the scene.

The CCSC for x-bands (polygon ID, edge list) {

- Store each polygon edge in F- and the B-sectors including its end points.
- Clear the F-, B-, and I-lists.
- Select the first F- and B-planes. /* They are radiated to -X. */
- While the F-plane is not the last /* The last is radiated to +X. */
  {
  - Swing the F-plane to the next position.
  - For each edge stored in the added F-sector {
    - if the edge is contained in the **F**-list,

      then { the edge is removed.}

      otherwise { the edge is added to both **F**- and **I**-lists.}

    }
  - Scan the **F**-list on the **F**-plane in order to compute polygon intersections, then trim away edge segments lying out of the x-band.
  - Scan the **B**-list on the **B**-plane and trim edges. /* As a result, the **I**-list refers to the edge segments contained inside the x-band.} */
  - Call the CCSC for y-bands with (polygon ID, the I-list, x-band ID).
  - Recover original positions of the trimmed edges.
  - Swing the B-plane to the next position.
  - For each edge stored in the removed B-sector {
    - if the edge is not included in the **B**-list,

      then {the edge is added.}

      otherwise {the edge is removed from both **B**- and **I**-lists.}

    }
  - Polygons stored in each section are sorted in increaseing order of teir Zmin.

}

Fig.4.5 Outline of Coupled Cylindrical Scan Conversion Algorithm

The CCSC algorithm for y-bands segments a polygon piece clipped by an x-band. It is the same algorithm as developed for the x-bands, however, the differences are:

(1) the **F**- and **B**-planes are perpendicular to the **Y**-**Z** plane and their projection centers are $\mathbf{S_p}$ and $\mathbf{S_m}$; the maximum and minimum y of the light source.

(2) In addition, the underlined statement in the previous algorithm is replaced by the following statements;

{

- Select the buffer cell related to the intersection of the x- and y-bands.
- Compute the shadow bounding volume of the clipped polygon.
- Store the polygon ID and the volume in the cell.

}

As a result, a polygon is memorized in the related cells of the ray-oriented buffer. After CCSC, polygons are sorted in each cell in increasing order of their z_{min} value.

4.3 Light Clipping

4.3.1 Cross Scanline Clipping

The cross scanline algorithm (cf. Appendix A) efficiently determines exact visible outlines by assuming polyhedral objects. We employ the algorithm to compute uncovered segments of a polygonal light source. Here the illuminated point is set to the view port, and the light center is the image center. The algorithm is named cross scanline clipping.

Since a polygonal light source radiates light rays to its front side, the plane containing the source splits the 3D space into an emitted and non-emitted side. Objects included in the non-emitted side are removed because they never cast shadows. If an object intersects the plane, the object is divided, then removed. Since illumination is computed at a point in the emitted side, each of the remaining objects must occlude the light source if they overlap. Therefore, uncovered light segments result in regions where no object exists except the light source. Note that cross scanline clipping does not need depth comparison, although the hidden surface removal algorithm does.

All polygons including a light source and object surfaces are segmented by **H**(horizontal) and **V**(vertical) scanlines. As shown in Fig.4.6, two **H**-scanlines are

allocated to the top and the bottom of the light source. **V**-scanlines are limited to lie between the **H**-scanlines. The first **V**-scanline is placed at the left end of the light source and the last at the right. The **H**- and **V**-scanlines define a rectangle region in which light clipping is executed. The other **V**-scanlines are located in the rectangle at polygon vertices, intersections of polygon edges and **H**-scanlines, and cross points of polygon edges. The cross points are efficiently detected during edge sort on a **V**-scanline. A summary of cross scanline clipping is given below.

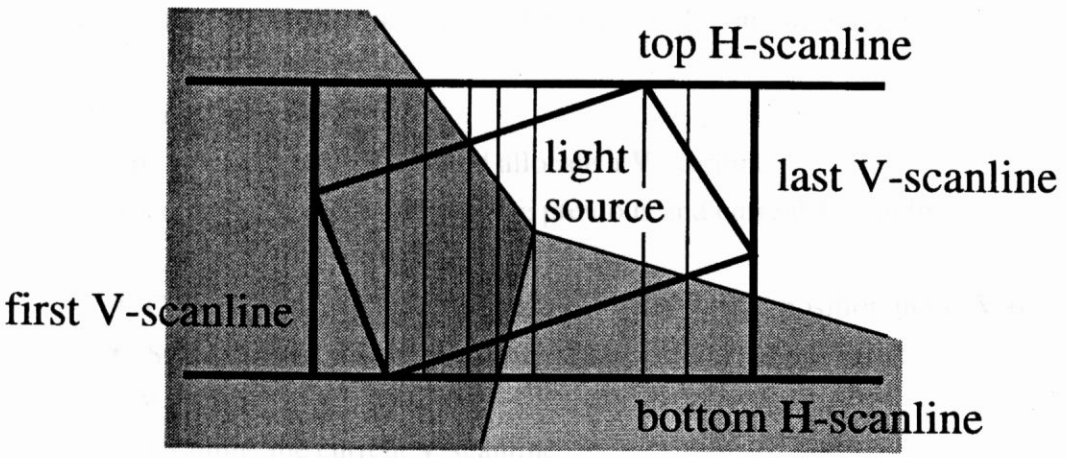


Fig. 4.6 Cross Scanline Algorithm

- Convert all polygons (*i.e.* a light source and object surfaces) in perspective from the observation point.
- Between the first and the last **V**-scanlines, {
  - Compute intersections of polygon edges to the **H**-scanlines.
  - Collect vertices existing between the **H**-scanlines.

  }
- Sort the intersections and vertices,
  then push their positions to a **V**-list.
- Examine the first **V**-scanline in order to determine polygon intersections,
  then save it.
- While the **V**-list is not empty {
  - Pop a position from the **V**-list and allocate a **V**-scanline.
  - If edges cross each other between the previous and current **V**-scanlines,
    then {
    - Cancel the **V**-scanline and push back the current position in the **V**-list.
    - Sort the cross points, then push their positions in the **V**-list.

    } otherwise {
    - Examine the current **V**-scanline.
    - Segment the area between the **V**-scanlines.
    - Memorize uncovered light regions, if exist.

    /* The region is described with a pair of edges. */
    - Replace the previous **V**-scanline with the current.

    }

  }
- Merge the light regions

Fig.4.7 Outline of Cross Scanline Clipping Algorithm.

As we mentioned above, if a segment includes no object except the light source, the segment is memorized. Between adjoining **V**-scanlines, the segment is uniquely described by a pair of edges because the edges never cross in the span. The intersections of the edges and the **V**-scanlines are accurately computed with floating point operations. Thus, the shape of the light segment is exactly obtained. Finally the obtained light regions are merged to reduce the light integration cost which is proportional to the number of light edges.

4.3.2 Fast Silhouette Generation

Light sources are actually segmented by the silhouette edges of each object. However, a tessellated object contains many inside edges when it is projected. Thus, the inside edges must be removed to avoid unnecessary V-scanline examinations. Unfortunately, exact silhouette generation for a concave object is as expensive as ordinary light clipping because edge-to-edge intersections must be tested. Silhouettes of convex objects are much easily created, however, decomposition of concave objects to convex forms also needs much computing cost.

We propose a fast and simple silhouette generation method that suits cross scanline clipping. First, back faces are removed as in ordinary hidden surface removal algorithms. Each edge of a front-face polygon is then labeled with an arrow as shown in Fig.4.8. The arrow means that its right side is filled by an object. If an edge has two directed arrows, for instance edge **EF** in Fig.4.8, objects are continuous over the edge, so that it can be removed.

This algorithm misses some inner edges of concave objects. For example, edges **BC** and **CD** in Fig.4.8 persist. However, cross scanline clipping achieves a complete determination of unoccluded light shapes although the object is counted twice in the overlapped region **BCDH**. The excess edges incur unnecessary computing cost, but the cost is expected to be much less than the overhead incurred by complete silhouette generation. Thus we implemented this simple algorithm.

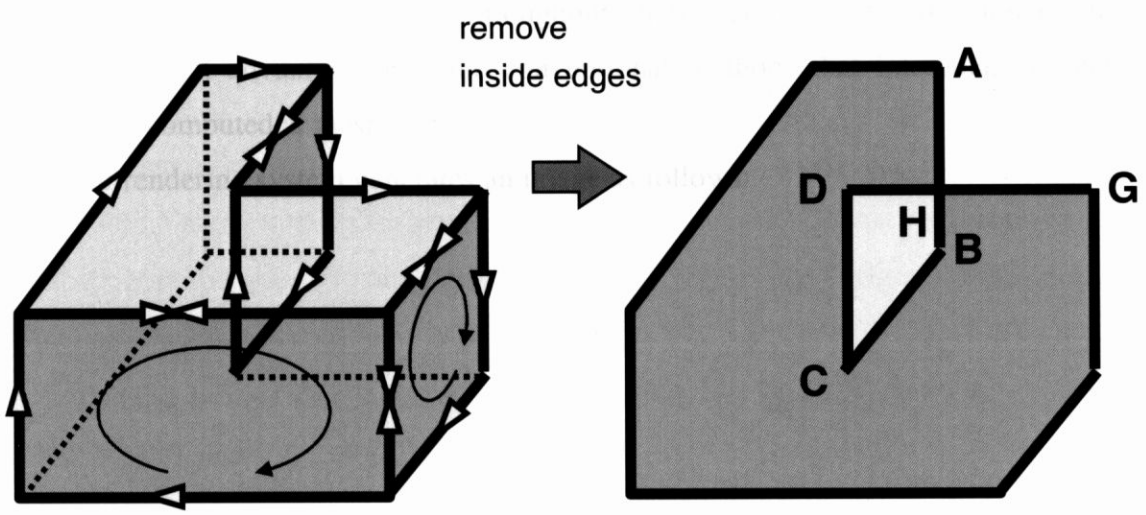


Fig.4.8 Silhouette edges.

4.4 Rendering Equation

We have already presented the rendering equation for area light sources that accurately computes both diffuse and specular reflections (see Appendix **B**). The equation assumes Lambertian distributive polygonal light sources and employs Phong's reflection model. Diffuse reflection is easily calculated by using the conventional method [Nishita83]. However, computation of specular reflection demands the integration of $cos^n \theta$ in the solid angle defined by an area light source and a rendering point on object surfaces. Here, n is the reflection index of Phong's model and θ is the angle between the light vector and the mirror reflection of the eye vector. The light vector is directed from the rendering point to the light source and the eye vector is to the observation point, so that angle θ varies during the integration.

The integration equation is precisely approximated with Chebyshev polynomials then analytically computed. Thus, the result is very reliable when generating 24 bit color images. In addition, the integration cost can be reduced by referring to adaptively sampled tables. Details of the integration techniques have been described in Appendix **B**.

Since sharp specular reflections such as highlights are very sensitive to positions of reflection points, our rendering algorithm computes the integration equation at every pixel, so that light clipping is needed at each pixel. As a result, our algorithm can represent exact specular intensity throughout all reflection levels from dull to sharp. This is a great advantage over the conventional methods that interpolate reflection intensities computed at mesh vertices.

Our rendering system generates an image as follows:

- Generate the ray-oriented buffer ;
  - Initialize the buffer.
  - Apply the *CCSC* algorithm to each polygon.
  - Sort polygons stored in each cell by z_{min} value.
- Determine the visible object at each pixel by using a hidden surface removal algorithm.
- For each pixel, {
  - Project the position onto the ray-oriented buffer to obtain a polygon list.
  - Select shadow casting polygons by using shadow bounding volumes.
  - Determine unoccluded light portions by cross scanline clipping.
  - Integrate both specular and diffuse reflection intensities.

  }

}

Fig.4.9 Outline of Rendering including Shadowing.

4.5 Experimental Results

We evaluated the proposed method using Pictures 4.1-4.5. The Teapot (Picture 4.1) and Chessmen (Picture 4.3) are samples of tessellated objects. The number of polygons in each scene is shown in Table 4.1. The Office* (Picture 4.2) was selected as a typical scene with a small number of large size polygons. The X29 (Picture 4.4) is an example of shadow projection onto an undulating surface. Size of the images was set to 640×480 pixels and buffer resolution was 128×128.

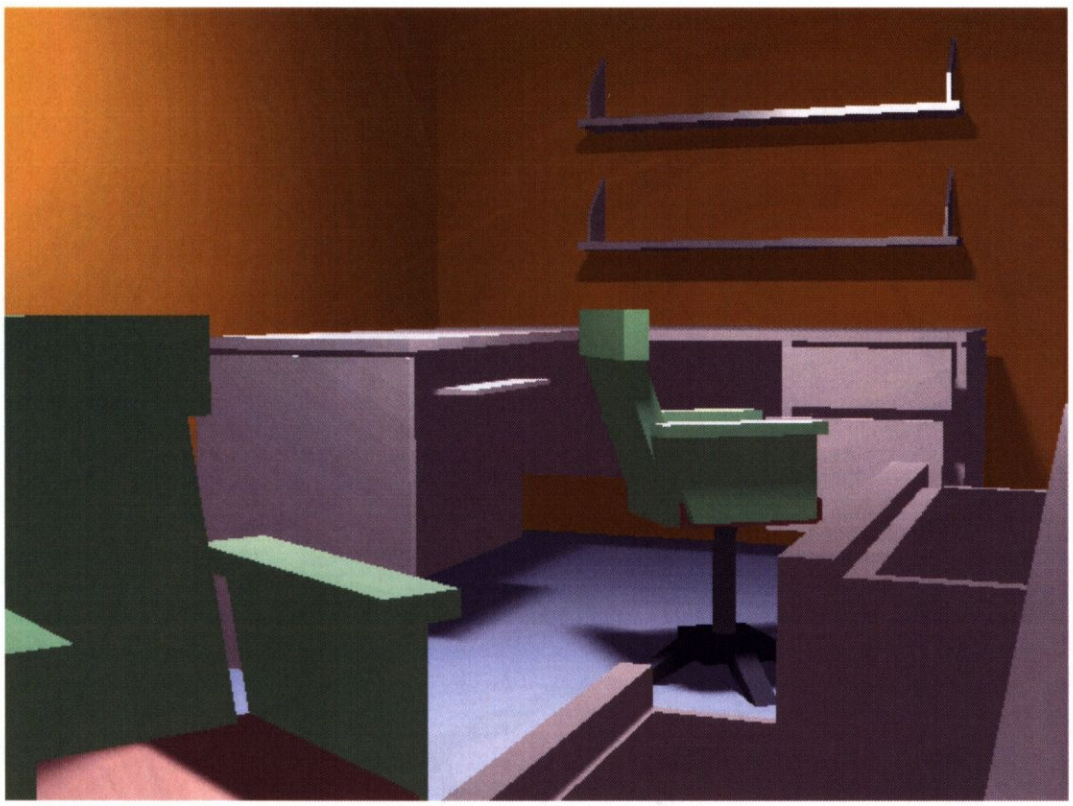
All images were generated on a SPARC station (60 MHz superSPARC+ with 88.9 SPEC 92 INT and 102.8 SPEC 92 FP).

* This data was presented by George Drettakis. The objects were used in his paper [Drettakis94], but were re-arranged by him. We changed the colors for test specular reflection.

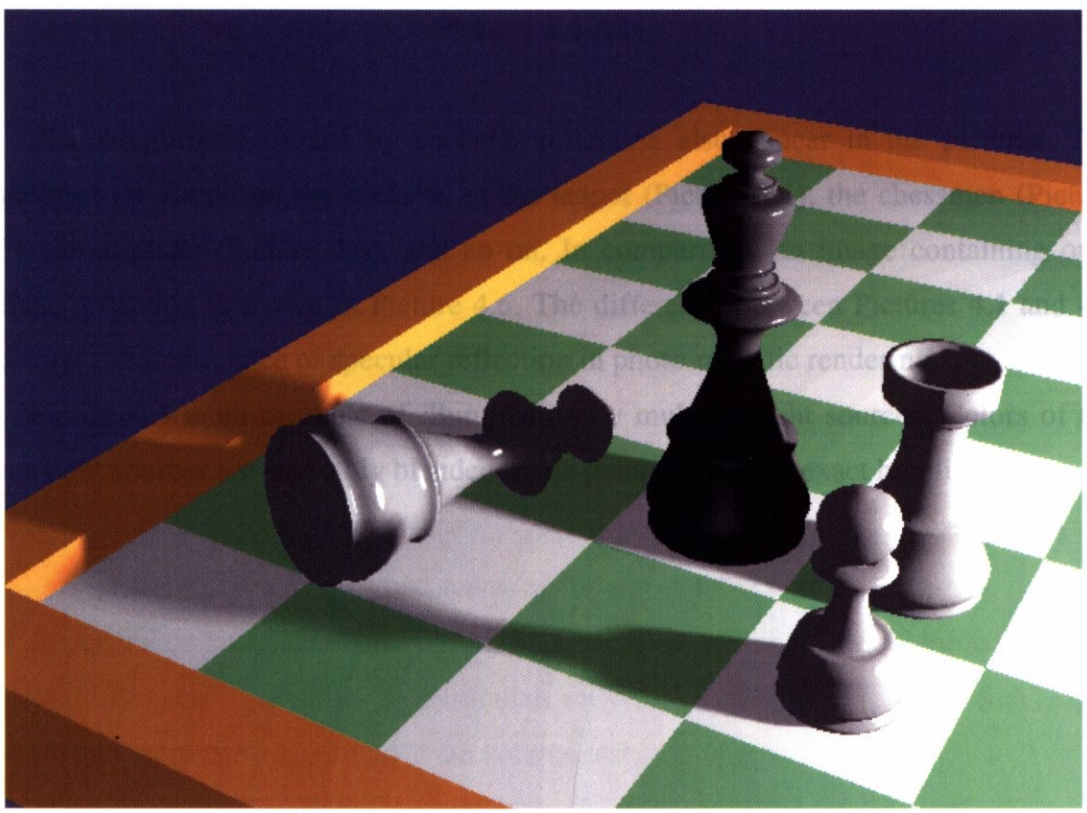
4.5.1 Image Quality

All images display realistic shadows. In Picture 4.1, a thin mesh drops dull shadows onto the tessellated surfaces of a teapot. The teapot also casts sharp shadows on itself and the floor. The chessman in Picture 4.3 successfully projects shadows onto the curved surfaces of other chessmen. In Picture 4.4, the airplane drops dull shadows onto an irregular ground constructed by cubes that are randomly pushed down or pulled up. The ground also drop shadows on itself. The intensity contrast of the umbras and penumbras yields improved image photo-realism.

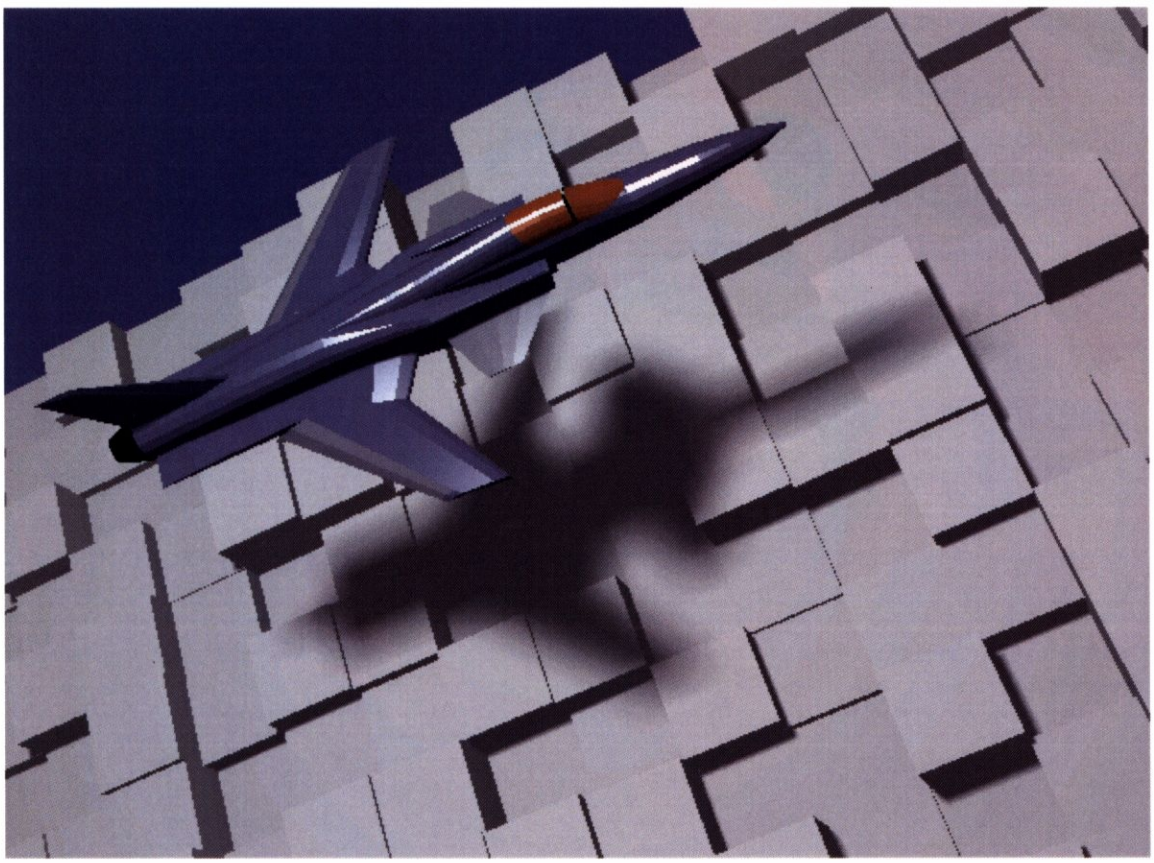
Picture 4.1 Teapot.



Picture 4.2 Office.



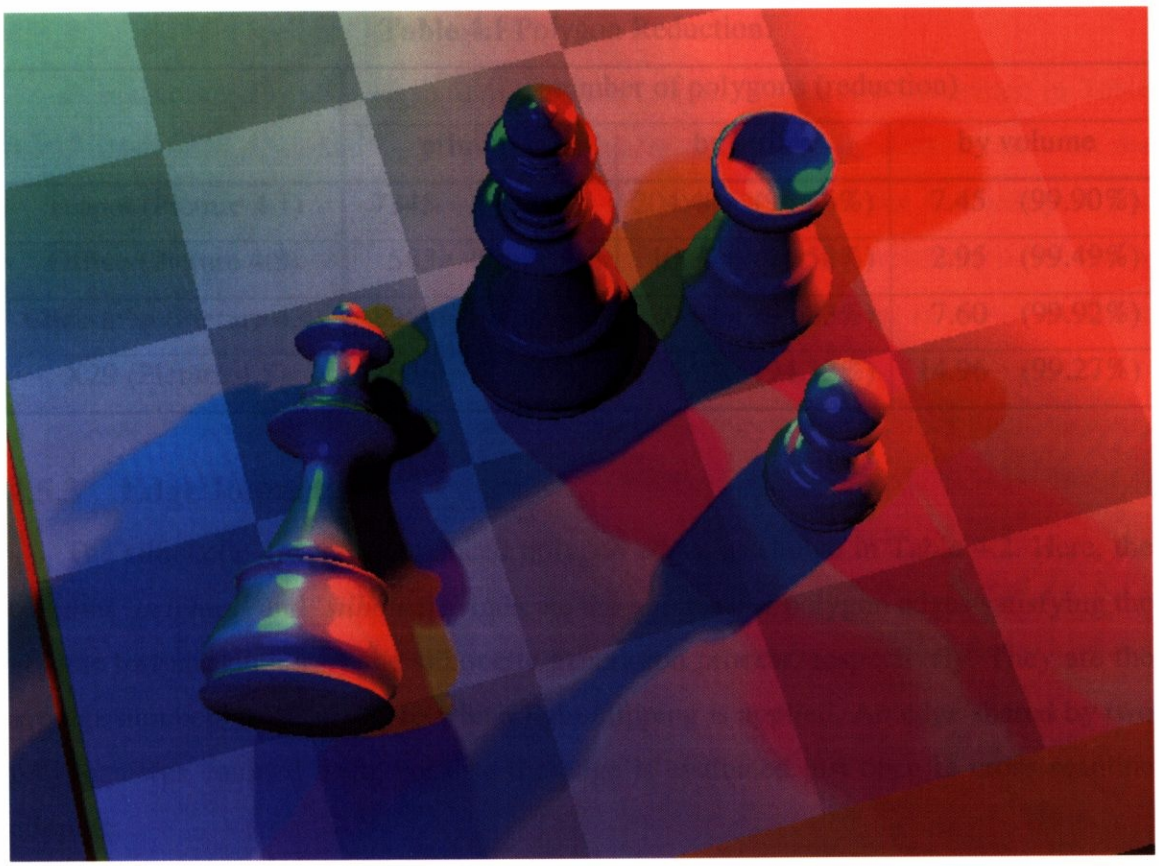
Picture 4.3 Chessmen.



Picture 4.4 X29.

Sharp highlights caused by specular reflection also appear in the pictures. The highlights are found on the surfaces of the teapot (Picture 4.1), the chessmen (Picture 4.3), the airplane (Picture 4.4), and so on. In comparison, an image containing only diffuse reflection is shown in Picture 4.6. The difference between Pictures 4.1 and 4.6 indicates the importance of specular reflection in photo-realistic rendering.

Picture 4.5 is an example of illumination by multiple light sources. Colors of the three light sources are smoothly blended in the penumbras with exact highlights.



Picture 4.5 Chessmen illuminated by multiple light sources.

4.5.2 Polygon Reduction

The ray-oriented buffer reduces the number of polygons considered in light clipping by two stages: the ray-oriented space subdivision and the shadow volume test. The reduction for Pictures 4.1-4.4 is shown in Table 4.1.

The column '*initial*' indicates the number of polygons included in the image data. If the buffer is not provided, light clipping must test the polygons at each pixel. The column '*space*' indicates the number of polygons remaining after space subdivision, *i.e.* stored in a buffer cell. The number of polygons clearing the volume test is shown as the '*volume*' value. These numbers are averaged for all rendered pixels.

Every image gives excellent results. Both the space subdivision and the volume test significantly reduce polygons. In particular, only 0.1 % of the polygons in the Teapot and Chessmen images remain after the volume test.

These experimental results demonstrate effectiveness of the ray-oriented buffer.

Table 4.1 Polygon Reduction.

	number of polygons (reduction)		
	primitive	by buffer	by volume
Teapot (Picture 4.1)	7345 (97.22%)	204.40 (96.36%)	7.45 (99.90%)
Office (Picture 4.3)	583 (98.22%)	10.37 (71.58%)	2.95 (99.49%)
Chessmen (Picture 4.4)	8958 (97.32%)	239.84 (96.83%)	7.60 (99.92%)
X29 (Picture 4.5)	2056 (87.60%)	254.97 (94.13%)	14.96 (99.27%)

4.5.3 Edge Reduction

The silhouette generation reduced polygon edges as shown in Table 4.2. Here, the columns '*original*' and '*silhouette*' indicate the numbers of polygon edges satisfying the volume test and output by the silhouette generation process, respectively. They are the average numbers in all pixels to which light clipping is applied. An edge shared by two polygons was counted once, because the edge is evaluated just once in cross scanline clipping.

A maximum of 46 % of polygon edges were removed. Since light clipping cost is related to the number of edges, the silhouette generation is very effective. However, the reduction was less than 10 % in the Office. This is because the Office scene contains a small number of large polygons. Thus the low reduction ratio is not an obstacle.

Table 4.2 Edge Reduction.

	number of edges		
	polygon	silhouette	reduction
Teapot	29.76	20.14	32.35%
Office	14.64	13.40	8.46%
Chessmen	31.66	17.12	45.94%
X29	78.83	40.49	48.64%

4.5.4 Image Generation Speed

Computing times for each module were measured. The results are written in Table 4.3. Buffer generation was very fast. The highest cost was only 2.08 seconds on the 60MHz SPARC station for 7345 polygons. This result proves the efficiency of the *CCSC* algorithm. The cost for examining the shadow bounding volumes is relatively cheap, although over a hundred polygons were tested at each pixel as shown in Table 4.1. This is because the shadow volume test is very simple.

Most of the computation required (about 50 %) was occupied by light clipping itself. The cost is logically related to the number of silhouette edges. This trend is recognized by comparing the '*clipping*' in Table 4.3 to the '*silhouette*' in Table 4.2. The image containing more silhouette edges requires longer clipping time except the X29. Of course it is not proportional because clipping cost depends on also object complexity.

Shadowing cost in Table 4.3 includes intensity integration cost for specular reflection at all pixels. However, the cost is relatively small because the integration is accelerated by referring to adaptively sampled tables. Since the tables promise sufficiently small approximation errors, no sampling artifacts appear in the resulting images. The table integration technique has already been presented in [Tanaka93].

As a result, each of the 4 images was generated in one or two minutes. Picture 4.5, the most complicated scene illuminated by three light sources, was generated in only 6 minutes and 21 seconds.

Table 4.3 Computing Cost.

	total (sec)	shadowing (sec)			shading (sec)
		buffer	volume	clipping	
Teapot	146.16	2.08	29.91	72.94	31.40
Office	49.93	0.78	8.48	23.60	12.85
Chessmen	9437	1.85	27.54	44.87	11.57
X29	200.02	3.15	36.74	108.76	45.57

4.5.5 Buffer Resolution

Higher resolution decreases the cost of the volume test because fewer polygons are saved in each buffer cell. However, buffer generation cost is proportional to buffer resolution. To evaluate their contribution, we varied buffer resolution for the Office and Chessmen images. The results are shown in Table 4.4. As expected, the buffer creation cost and the volume examination cost are sensitive to buffer resolution.

The light clipping cost slowly decreases as the buffer resolution increases. This is because higher resolutions more severely eliminate false polygons. However, only 20 % of the light clipping cost was reduced while the buffer size was enlarged from 32×32 to 256×256. This indicates that the shadow bounding volume is a very good approximation of the real shadow space. Conversely, the shading cost is constant as is logical. These results confirm this fact.

Higher resolution reduces the image generation cost, but not drastically. The low sensitivity of cost to buffer resolution is a strong point of our method.

Table 4.4 Buffer Resolution.

	buffer resolution	total (sec)	shadowing (sec)			shading (sec)
			buffer	volume	clipping	
Office	32×32	53.11	0.23	8.66	27.46	13.04
	64×64	51.50	0.22	8.55	25.91	13.01
	128×128	49.93	0.78	8.48	23.60	12.85
	256×256	49.90	2.83	7.73	22.41	12.98
Chess-men	32×32	150.52	0.96	76.95	52.57	11.62
	64×64	112.71	1.13	44.35	46.81	11.90
	128×128	94.37	1.85	27.54	44.87	11.57
	256×256	89.74	3.90	23.34	42.67	11.40

4.6 Conclusion

This chapter has described a rendering algorithm that generates photo-realistic images containing penumbras and highlights from area light sources. For accurate shading and shadowing, exact determination of unoccluded light portions is essential. The proposed method achieves fast and exact light segmentation by combining a ray-oriented buffer and cross scanline clipping.

A ray-oriented buffer subdivides 3D space by following light rays and saves polygonal objects intersecting each subspace. Bounding volumes of shadow spaces caused by each polygon are also saved in the buffer. The space subdivision and the volumes together reduce the number of polygons subjected to clipping by eliminating polygons that not occlude the light sources. Experimental results confirmed that less than 1 % of all polygons were subjected to clipping. The buffer was efficiently constructed by using the CCSC algorithm presented in this chapter.

Cross scanline clipping splits an area light source with horizontal and vertical scanlines in order to determine unoccluded light segments efficiently. We also proposed a fast silhouette generation algorithm that suits cross scanline clipping.

We employed our previous rendering equation for Lambertian distributive area light sources. It accurately integrates both diffuse and specular reflection using Phong's model. A sophisticated image can be created in a few minutes by the combination of the ray-oriented buffer, the cross scanline clipping, and the rendering equation even if objects cast shadows onto curved or rugged surfaces. Its speed is very practical for fine rendering.

The algorithms and techniques proposed here can be extended to global illumination. Cross scanline clipping is applicable to analytic form factor solutions in radiosity [Cohen83]. A combination of clipping and ray-oriented buffers would suit fast discontinuity mesh generation. Light sources with complex distribution properties can be rendered by adding an advanced general illumination model such as Arvo's [Arvo95], although reflection integration would become much more complex. They are future works.