國立臺灣大學電機資訊學院資訊工程學研究所

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

利用屏障圖產生互動性全域照明

Interactive Global Illumination by Occlusion Map

吳榮軒

Wu, Jung-Hsuan

指導教授：莊永裕 博士

Advisor: Chuang, Yung-Yu, Ph.D.

中華民國 98 年 7 月

July, 2009

國立臺灣大學
資訊工程學研究所

碩士論文

利用屏障圖產生互動性全域照明

吳榮軒 撰

98
7

國立臺灣大學碩士學位論文
口試委員會審定書

利用屏障圖產生互動性全域照明

Interactive Global Illumination by Occlusion Map

本論文係吳榮軒君 (R96922056) 在國立臺灣大學資訊工程學研究所完成之碩士學位論文，於民國 98 年 7 月 15 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

____

____

____

____

____

所 主 任 ______________________________

中文摘要

我們提出一種適用於動態場景，動態視角，及動態光源的全域照明演算法， 我們為螢幕上的每個像素產生出一張屏障圖，將這張屏障圖當做間接反射的 光源，就可以計算出全域照明的效果，包含柔和陰影，反射照明，以及多次 反射照明等效果，都可以藉由遞迴式的渲染程序合成出來。我們模擬了傳統 的渲染方程式並將之加速到互動速率。

Abstract

We present an algorithm for rendering global illumination while the scene, camera, and light sources can be fully dynamic. We achieve this by generating an occlusion map of each pixel on screen and take the occlusions as indirect light sources to produce the global illumination effects. By exploiting the occlusion map, global illumination effects including soft shadow, reflective lighting, and multi-bounce lighting can be produced by iteratively rendering process. We have approximated the rendering equation and accelerate it to interactive rate.

Contents

口試委員會審定書 **i**

中文摘要 **iii**

Abstract **v**

1 Introduction **1**

2 Related Work **3**

3 Occlusion Map **5**

3.1 Sampling direction map generation 5

3.2 Occlusion map generation 6

4 Rendering **7**

4.1 Direct illumination 7

4.2 First bounce indirect illumination 7

4.3 Multi-bounces indirect illumination 8

5 Acceleration **9**

5.1 Scale down framework 9

5.2 Scale up framework 10

5.3 Rendering acceleration 11

6 Experiments and Results **13**

6.1 Performance 13

6.2 More results 13

7 Conclusion and Future Work **23**

Bibliography **25**

List of Figures

3.1 Left: The ray r started from pixel p hits triangle t. Right: The rays r started from the pixels in patch a might hit triangle t_i and t_j 6

6.1 Helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively. 14

6.2 Helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively. 15

6.3 Venus and two helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively. 16

6.4 Venus and two helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively. 17

6.5 A series of images showing that the scene, lighting, and the camera are all changing. The left column shows the direct lighting only. The middle column shows the indirect lighting only. The right column shows the global illumination effects which is the combination of the images on the left column and middle column. 18

6.6 A series of images showing that the scene, lighting, and the camera are all changing. The left column shows the direct lighting only. The middle column shows the indirect lighting only. The right column shows the global illumination effects which is the combination of the images on the left column and middle column. 19

List of Tables

6.1 Rendering helicopters using different parameter settings 20
6.2 Rendering Venus and helicopters using different parameter settings 21

Chapter 1

Introduction

Global illumination can produce many important effects including soft shadows, inter-reflections, etc. . Therefore, it is utilized in most animations, movies and video/PC games to render realistic scenes. However, computing precise global illumination is very time-consuming such that some approximation algorithm should be proposed in practical. The main idea of global illumination effects is that a pixel is lit not only by light sources but also by other surfaces. Lights can be reflected from other surfaces even if those surfaces emit no lights themselves. To compute the illumination of one pixel needs to sum up reflected lights from all other objects. To render a scene, such computation must be applied to all pixels on the screen and the time complexity is positively related to the square of the number of polygons of the scene.

Although the computational power grows exponentially, the scenes used nowadays are constantly more complicated and detailed. Rendering precise global illumination effects in real-time is still not achievable. Much research has been turned back to obtain an approximation to produce a visually correct result rather than a physically correct one. As discussed by Kozlowski et al.[12], the knowledge about visual system and its perception of materials [11][1][7][8] and shapes [2][9] have been utilized to reduce the unnecessary computation in many real-time and interactive rendering algorithms for efficient image synthesis. The ignorable part is often determined by its visual significance. For example, since the more times that a light reflected, the less energy which the light can contribute, the computation of the direct lighting is more important than the indirect lighting in most cases. To preserve the computation of the most important effects, Wang et al.[18] develops an algorithm to render the dynamic scenes under direct illumination only. Dachsbacher and Stamminger [3] and Ritschel et al.[14] extend the lighting to the first bounce indirect illumination. The work of Dachsbacher et al.[4] allows the trade-off between the computing time and the reality of the synthesized image.

In this paper, we propose an approach to render the first and multi-bounces indirect illumination. We estimate occluders from every point visible by the camera to every direction sampled by importance, and utilize these occluders as indirect light sources to render first bounce indirect lighting. For multi-bounces indirect lighting, we approximate it by only bouncing lights between points visible by the camera, which is also used in many image-based approaches. We also propose a framework to speed up the process of estimating occluders for every pixel. And our results show our approach works with acceptable visual quality and interactive frame rate.

Chapter 2

Related Work

GPU ray tracing: tracing rays by exploiting the power of the GPU is a commonly used skill in graphics. Due to the large number of processing units in a GPU and high parallelizability of the ray tracing algorithm, the basic ray tracing has already been able to finish in less than one second for scenes which is not too complex. However, the time cost is positively related to the complexity of the scene, so it still can not afford to render a scene composed of tens of thousands of triangles. To reduce the time cost, many algorithms are developed, such as bounding box, octree, binary partition tree, etc. . We introduce a method by exploiting the GPU and tracing rays to generate occlusion maps for rendering indirect illumination effects.

Ambient occlusion: for each pixel, the illumination intensity can be determined by the average visibility direction and its total spherical area. The occluder proximity can then be utilized to produce maximally soft shadows. Our method is similar to the works of Shanmugam and Arikan[15] and Dimitrov et al.[5] which are also screen-space based methods. The calculations on the pixels which are not visible from the camera will not be involved and thus can save time. However, the effects produced by the invisible geometry will lost in this way, so we try to preserve the information of the hidden geometry by generating occlusion map only for the pixels visible from the viewpoint.

Illumination from many lights: global illumination effect can be thought as a scene lit by many indirect light sources. The points on the scene hit by the rays emitted from the light sources will become the new light sources for rendering the scene with one more bounce lighting in the next iteration. Several methods try to reduce the cost which grows with the number of light sources, Walter et al [17] introduced Lightcuts, a scalable solution for handling many lights by a hierarchical algorithm, and Hašan et al.[10] do sampling and clustering with the GPU and improve the speed into a few seconds per frame. Our method also uses many lights method for indirect lighting, but we do not assume that all light sources, especially indirect light sources, are point light sources. We also consider about BRDFs of materials of indirect light sourcesthe for rendering a more realistic image.

Precomputed Radiance Transfer: Precomputed Radiance Transfer (PRT)[16][13] allows real-time rendering of some complex global illumination effects, such as soft-shadows and diffuse/glossy interreflections. PRT first parameterizes the incident lighting over the whole scene by means of different basis functions, such as spherical harmonics and wavelets, and compute and save the solution. The precomputed solution will then be referred in rendering process. However, it is often limited to a static scene and low frequency lighting, and requires a large amount of storage of the solution and expensive precomputation involving inter-object ray tracing. Recently, the direct illumination of deformable models and high-frequency lighting can be handled by different PRT techniques. Unfortunately, there are still some effects including interreflections and self-shadowing cannot be rendered in this setting. The reason is that the visibility is almost impossible to be reproduced by any precomputations. Our algorithm tries to reproduce the reflective lighting over dynamic scenes by computing the up-to-date visibility every frame because it is the only way.

Chapter 3

Occlusion Map

To render a scene with global illumination, we need to find triangles which can light the pixels on screen. With a visibility vector of every pixel to all triangles, we can calculate the indirect lighting I_p on the pixel p by summing up the contribution from every visible triangle:

$$I_p = \sum_{i=1}^{n} I_{t_i} * V_{p,t_i}$$

Notice that I_{t_i} is the light contributed by triangle t_i, V_{p,t_i} is the visibility between the triangle t_i and the pixel p, and n is the number of triangles. In spite of finding the visibility between all triangles and a specific pixel, we generate the visibility vector by doing importance sampling on the hemisphere centered at the pixel according to the BRDF of the material. For each sampling direction, a ray is emitted from every pixel along the sampling direction and the position of the nearest intersected point will be saved. The intersected points will be taken as indirect light sources for every pixel when rendering global illumination effects. By using a direct lighting map rendered with original light sources, the first bounce indirect lighting to all pixels on the screen can be calculated and utilized to generate the second bounce indirect lighting and so on.

3.1 Sampling direction map generation

Our occlusion map is generated base on importance sampling. Therefore, we should first determine and store chosen sampling directions of each pixel in another map, which is called as a sampling direction map. Each element of the sampling direction map, $D(p, j)$,

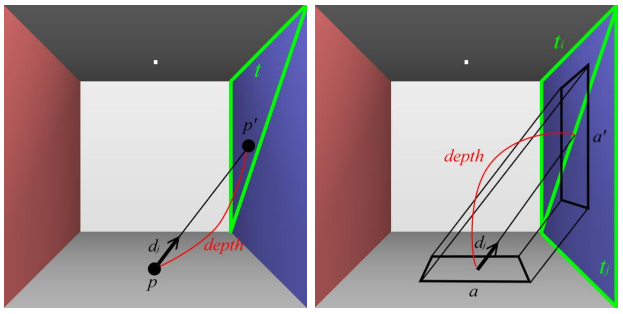


Figure 3.1: Left: The ray r started from pixel p hits triangle t. Right: The rays r started from the pixels in patch a might hit triangle t_i and t_j

is a normalized vector where j is the index of the random sample given by importance sampling according to the BRDF of pixel p.

3.2 Occlusion map generation

Every element in the occlusion map, $M(p, d_j)$, is the nearest position p' in the scene to the pixel p along the sampling direction d_j as shown on figure 3.1 (left). We utilize the depth buffer supported by graphics hardwares to find the nearest position in the scene hit by the ray emitting from pixel p. Notice that only the position of the least depth value will be saved and thought as one of the indirect light sources for the pixel p. Algorithm 1 shows the steps to generate an occlusion map.

Algorithm 1 Generate an occlusion map

```
for each pixel $p$ do
  for each random sampled direction index $j$ do
    ray $r$;
    $r.pos$ = position of $p$ in world space;
    $r.dir = D(p, j)$;
    if $r$ hits any triangle $\in triangle\_list$ then
      $M(p, r.dir)$ = the nearest point hits by $r$;
    end if
  end for
end for
return $M$
```

Chapter 4

Rendering

Now the positions of the indirect light sources of every pixel are stored in the occlusion map, $M(p, d_j)$. However, we still need to know the intensities of the indirect light sources to render the indirect lighting effects.

4.1 Direct illumination

Intensities of the indirect light sources for generating the first bounce indirect lighting effects are estimated by rendering direct lighting in light space first. We first render the scene with direct lighting in light space on an texture (L_{direct}) with the alpha channel saving the depth value in light space. Here we use the work of Donnelly et al. [6] to generate the direct illumination with soft shadows.

4.2 First bounce indirect illumination

We can then obtain intensities of the occluders, which will be taken as indirect light sources, by inquiring the direct lighting map, L_{direct}. Notice that not all indirect light sources can be lit by the original light sources because some of them might be in shadows of other objects. The problem can be easily solved by using the shadow map proposed by Lance William [19] and the intensity of those who are not lit will be zero. Given the intensities of all indirect light sources, the intensity of a pixel pcan be calculated by multiplying the cosine value of the angle between the normal vector of the pixel pand the incoming direction of each indirect light sources, $M(p, d_j)$, and sum them up. Algorithm 2 shows the steps to render the first bounce indirect illumination.

4.3 Multi-bounces indirect illumination

We assume lights only bounce between points visible by the camera. Therefore, we can replace the direct lighting map, L_{direct}, in algorithm 2 by an indirect lighting map to estimate the next bounce indirect lighting map. Note that we also change to transform $M(p, d_j)$ into the screen space instead of the light space when we estimate p_d. It is because indirect lighting maps are described and indexed in the screen space.

Algorithm 2 Render the indirect lighting map, $L_{indirect}$

```
Initialize $L_{indirect} = 0$;
**for** each pixel $p \in L_{indirect}$ **do**
  **for** each random sampled direction index $j$ **do**
    $p_d$ = transforming $M(p, d_j)$ into light space;
    $\omega_i = normalize(M(p, d_j) - p.pos)$;
    $\omega_j = normalize(camera.pos - p.pos)$;
    $L_{indirect}(p)$ += $L_{direct}(p_d) * BRDF(p_d, \omega_i, \omega_j)/$
              $length(M(p, d_j) - p.pos)^2$;
  **end for**
**end for**
return $IL$
```

Chapter 5

Acceleration

5.1 Scale down framework

Due to the large amount of pixels on the screen, the time complexity will be so huge that this process might not be done in interactive rate. So we must modify previous algorithms to apply to a patch a on the screen rather than to each pixel, as shown on figure 3.1(right). A patch is combined with the pixels which are similar enough to each other, which means that those pixels have similar positions in world space and similar normal vectors, so that they will very possibly have similar value on the generated occlusion map. For the similarity of the position and sampling directions, we define the similarity of any two pixel p_i and p_j as:

$$isSimilar(p_i, p_j) = \begin{cases} 1, & \text{if } dot(p_i.nor, p_j.nor) > \xi_{nor} \text{ and} \\ & abs(p_i.dep - p_j.dep) > \xi_{dep} \text{ and same BRDF} \\ 0, & \text{otherwise.} \end{cases}$$

Notice that ξ_{nor} and ξ_{dep} are the thresholds of normal and depth, respectively. Besides, the depth here is not the distance from the pixel to the scene used in the generation of the occlusion map but the depth value (z value) in camera space. We use the similarity to scale down the position buffer PB and the normal-depth buffer NDB:

$$PB(p_s) = \frac{1}{m} \sum_{p \in patch} p.position$$

$$NDB(p_s) = \frac{1}{m} \sum_{p \in patch} p.normal\_depth$$

where p_s is the position of pixel p after scale down, and m is the number of pixels which is similar to each other in the patch. The down-scaled occlusion map can then be calculated by using the down-scaled position buffer PB and normal-depth buffer NDB.

Besides, a patch might hit more than one point and even more than one triangle on the sampling direction, so the down-scaled occlusion map will not save the nearest position p but save the nearest several triangles. To make each element of the down-scaled map to save a list of the triangles, we first separate all triangles into k groups and assign the k groups of triangles to algorithm 1 and it needs some modification to return the index of the nearest triangle rather than the position. Then we can get k different occlusion maps and the element of the pixel p_s on the k maps is the list of the triangles which contain the candidates of the indirect light sources.

5.2 Scale up framework

To render the multi bounces indirect illumination effects, the list of the triangles generated from the down-scaled step will be assigned to the original algorithm 1 to obtain the position of the indirect light sources of the pixels in the original scale. However, the scaling down process only considers the pixels which are similar enough to each other. For those pixels which are not considered in, we build a reference map $Ref(p)$ which contains the vector from the corresponding pixel in down-scaled occlusion map to one of its neighbors which is similar with pixel p,

$$Ref(p) = \begin{cases} (0,0), & \text{if } isSimilar(p, np_s) = 1 \\ np_s - p_s, & \text{if } isSimilar(p, np_s) = 1 \text{ and} \\ & \exists np_s \in \text{ neighbors of } p_s \end{cases}$$

where p_s is the corresponding pixel of p on down-scaled occlusion map and np_s is one of the neighbors of p_s. And then the map saving the list of nearest triangles $M(p, d_j)$ can be built with the reference map $Ref(p)$ and the down-scaled occlusion map $M_s(p_s, d_j)$:

$$M'(p, d_j) = M_s(p_s + Ref(p), d_j)$$

5.3 Rendering acceleration

The calculation of the intensities which are contributed from all indirect light sources is also time-consuming. Because the indirect illumination is usually smooth, we can first render the indirect lighting from all indirect light sources to a smaller texture and then scale the final result up to required resolution by bilinear interpolation. In this way, we can reduce the cost of gathering the contribution from all indirect light sources.

Chapter 6

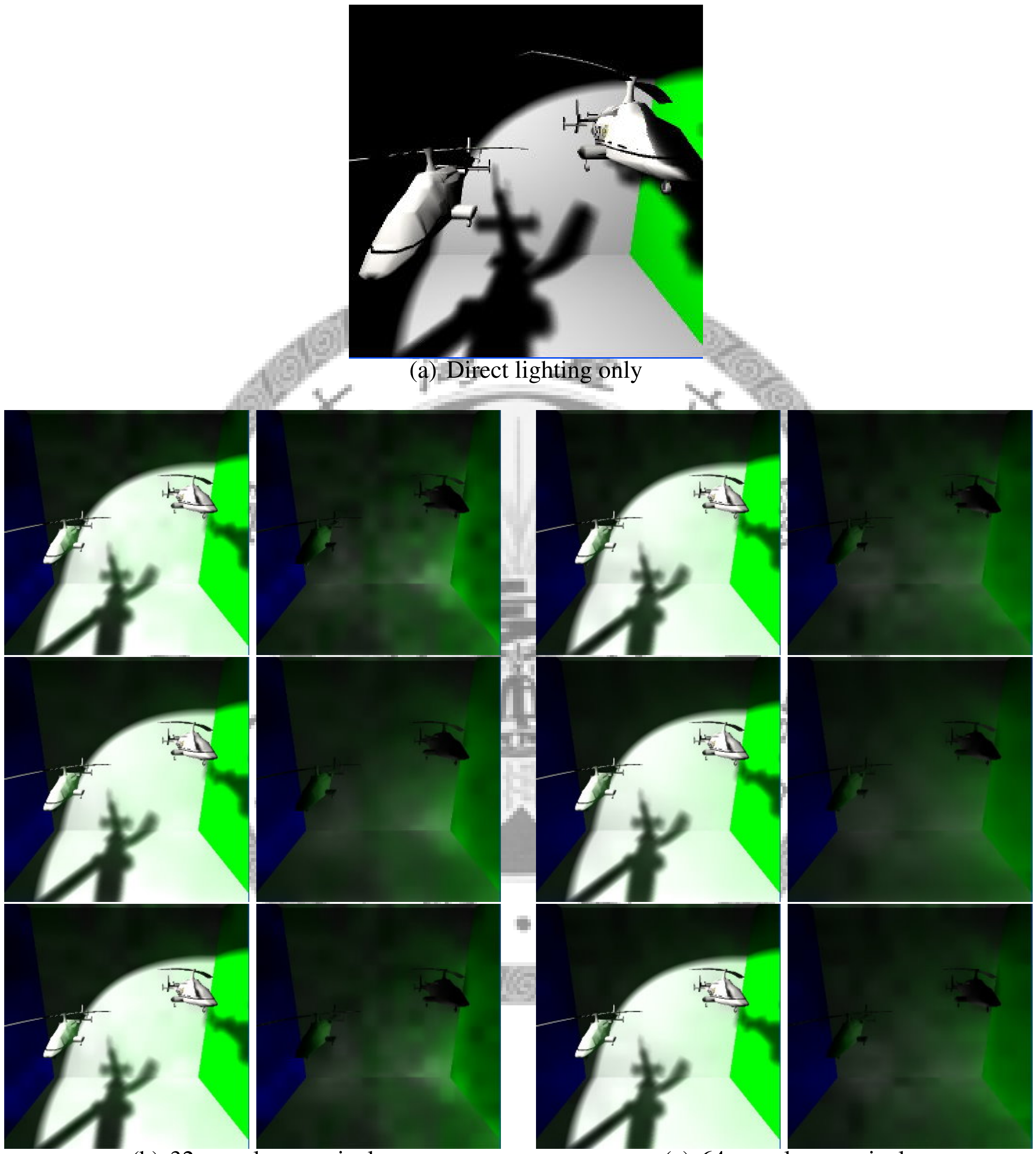
Experiments and Results

6.1 Performance

We have implemented our algorithm and execute it on a machine with Intel CPU Q6600 2.4GHz, 2GB ram and NVIDIA GeForce GTX 280. The resolution of all our results in this paper were set to 512*512. Since it is hard to balance between the performance and the quality of the out image, we have some parameters which can be adjusted by the users to decide what quality of the output animation they want in a reasonable frame rate. We list our experiments in Table 6.1 and 6.2 with several general parameter settings.

6.2 More results

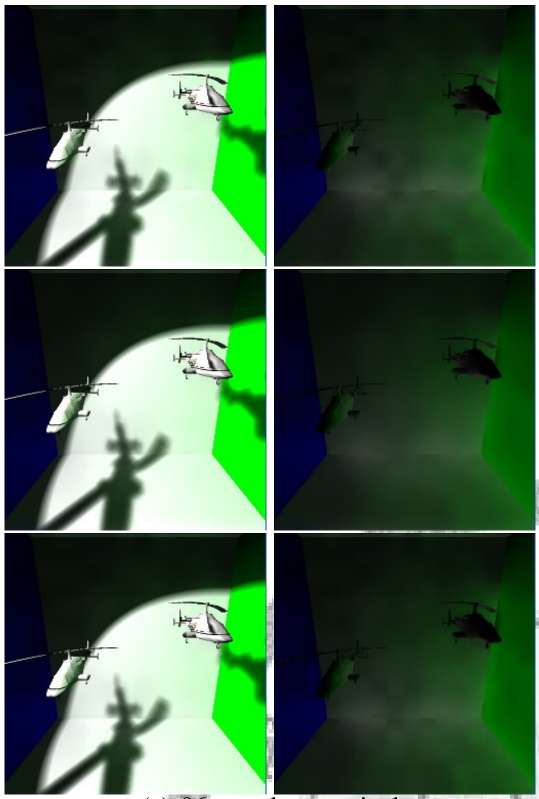
We have rendered an animation with the scene, the lighting, and the camera are all dynamically changing and took some screenshots Figure 6.5 and 6.6. We rendered this animation with 128 indirect light sources per pixel and the indirect illumination map was rendered in resolution 256*256 first. The rendering rate of the whole animation is about 5 fps.

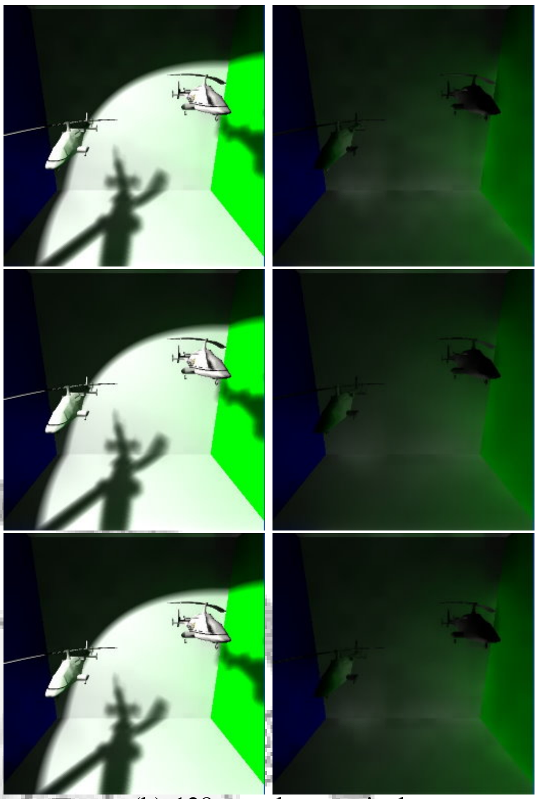

(a) Direct lighting only

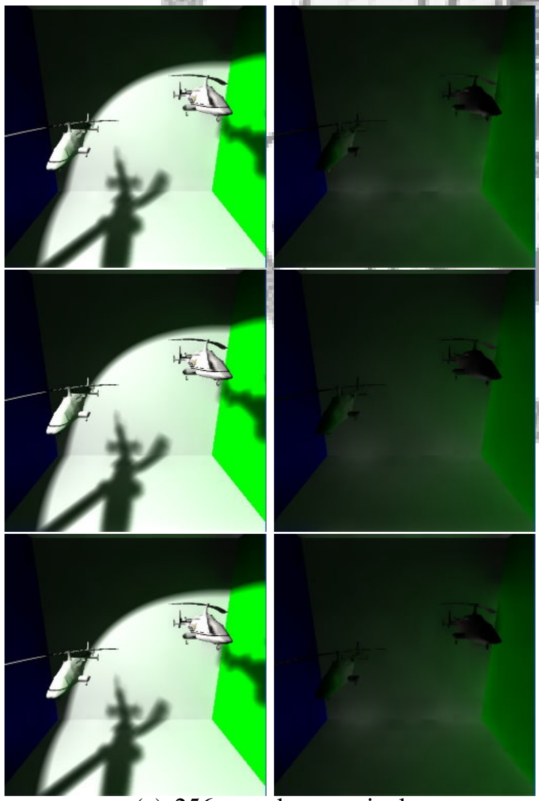
(b) 32 samples per pixel

(c) 64 samples per pixel

Figure 6.1: Helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively.


(a) 96 samples per pixel


(b) 128 samples per pixel


(c) 256 samples per pixel

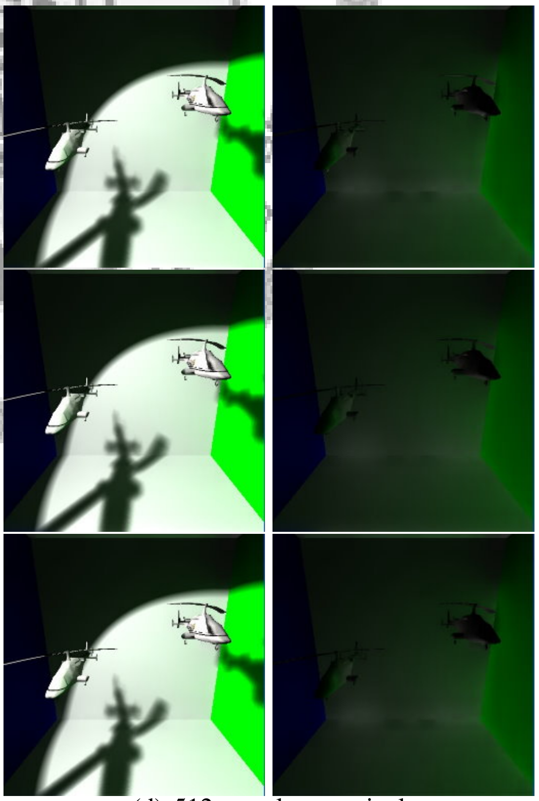
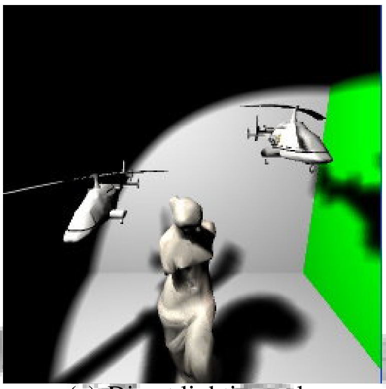
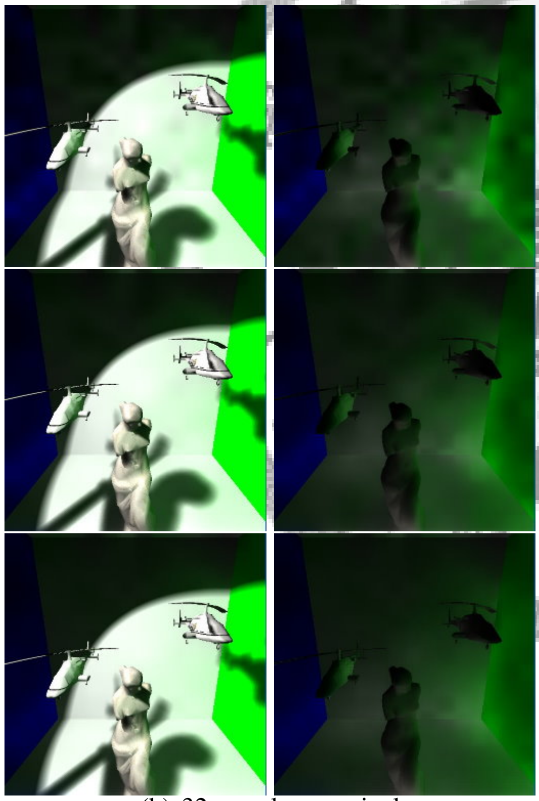

(d) 512 samples per pixel

Figure 6.2: Helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively.


(a) Direct lighting only


(b) 32 samples per pixel

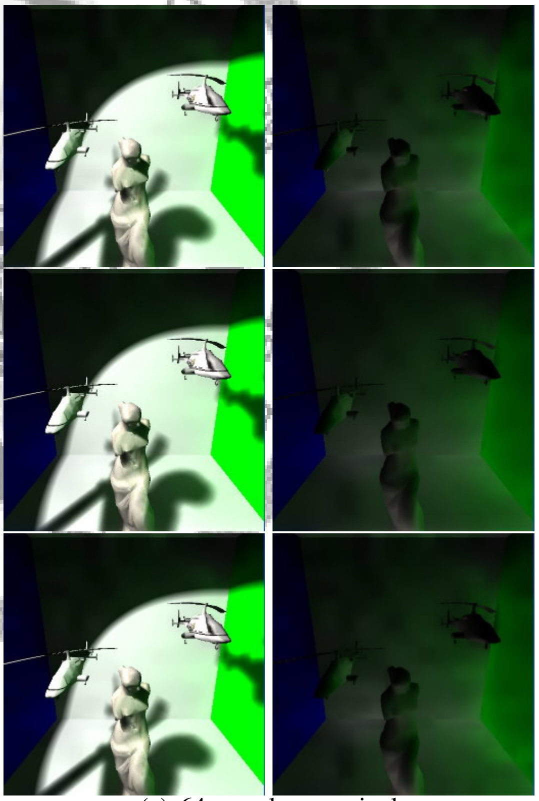
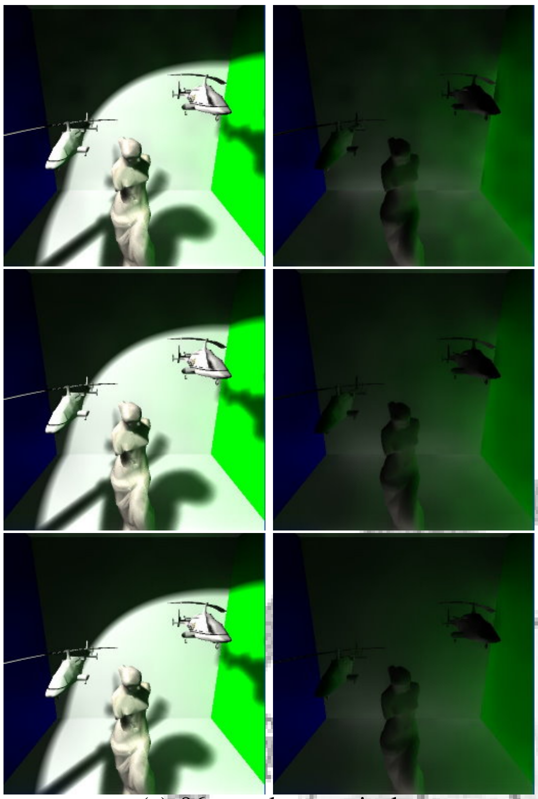
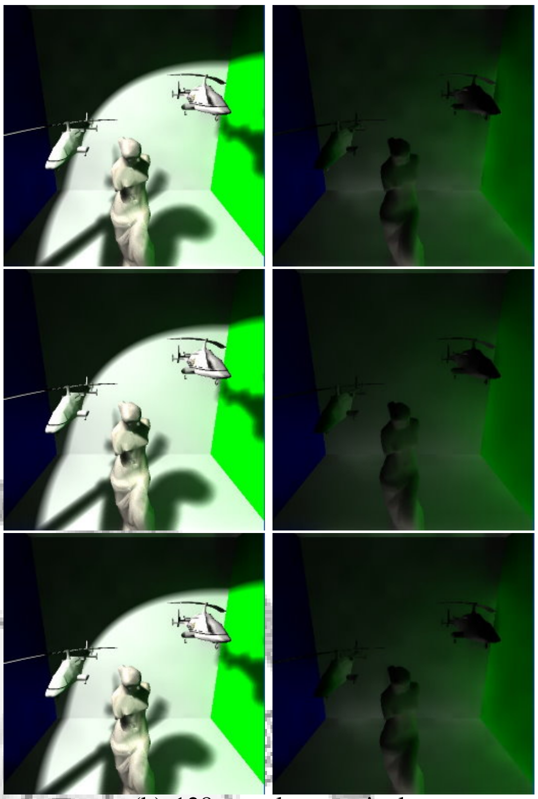
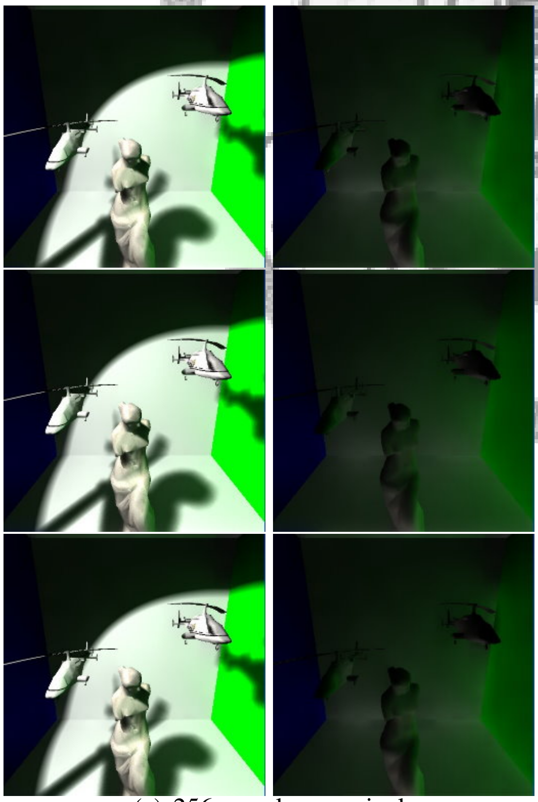

(c) 64 samples per pixel

Figure 6.3: Venus and two helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively.


(a) 96 samples per pixel


(b) 128 samples per pixel


(c) 256 samples per pixel

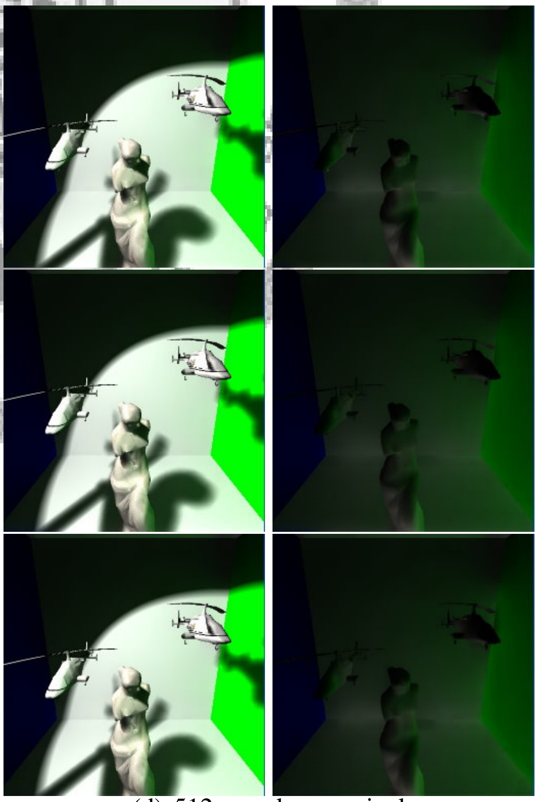

(d) 512 samples per pixel

Figure 6.4: Venus and two helicopters with different parameter settings, the resolution of the indirect illumination of the images on the first row, the second row, and the third row is 512*512, 256*256, and 128*128, respectively.

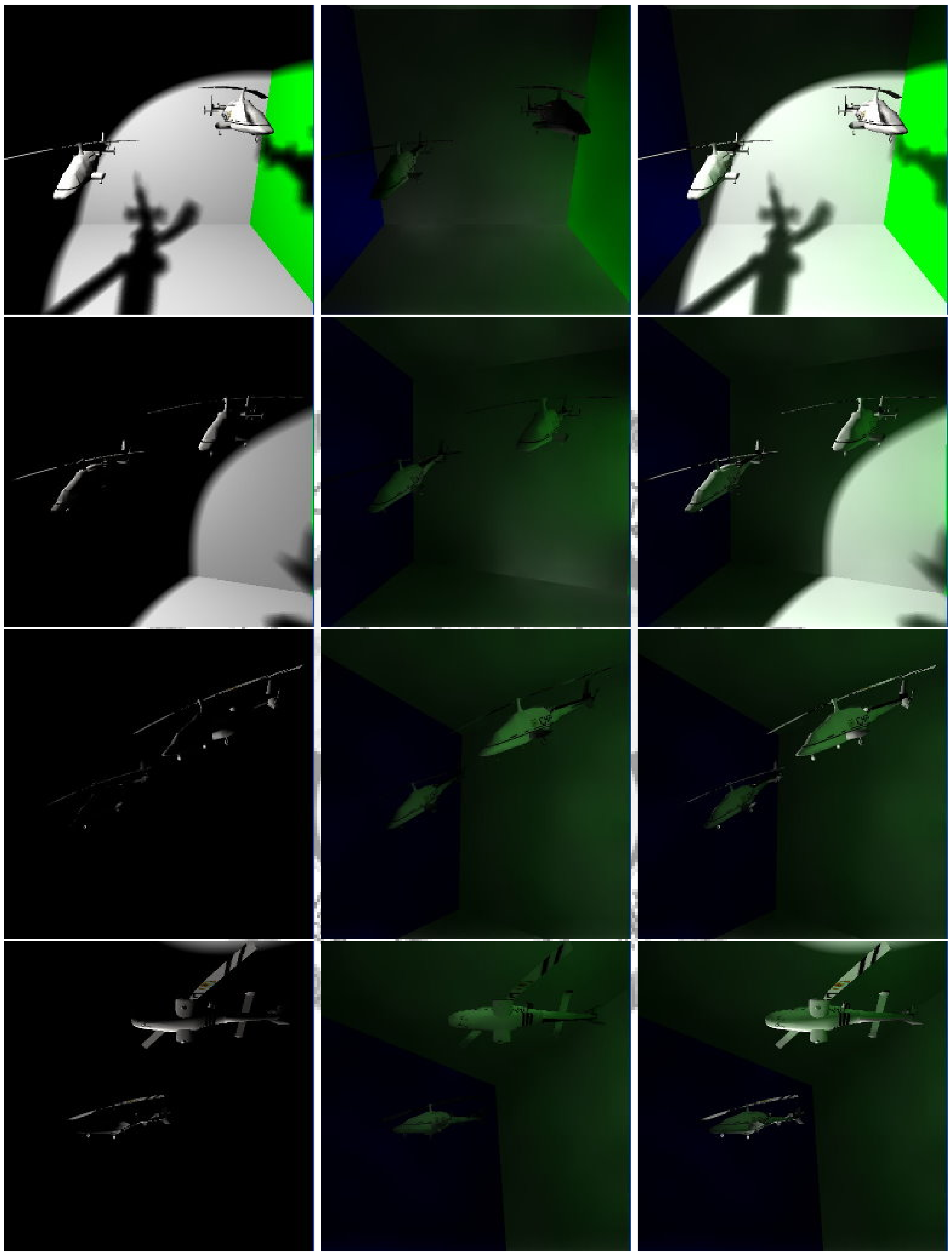


Figure 6.5: A series of images showing that the scene, lighting, and the camera are all changing. The left column shows the direct lighting only. The middle column shows the indirect lighting only. The right column shows the global illumination effects which is the combination of the images on the left column and middle column.

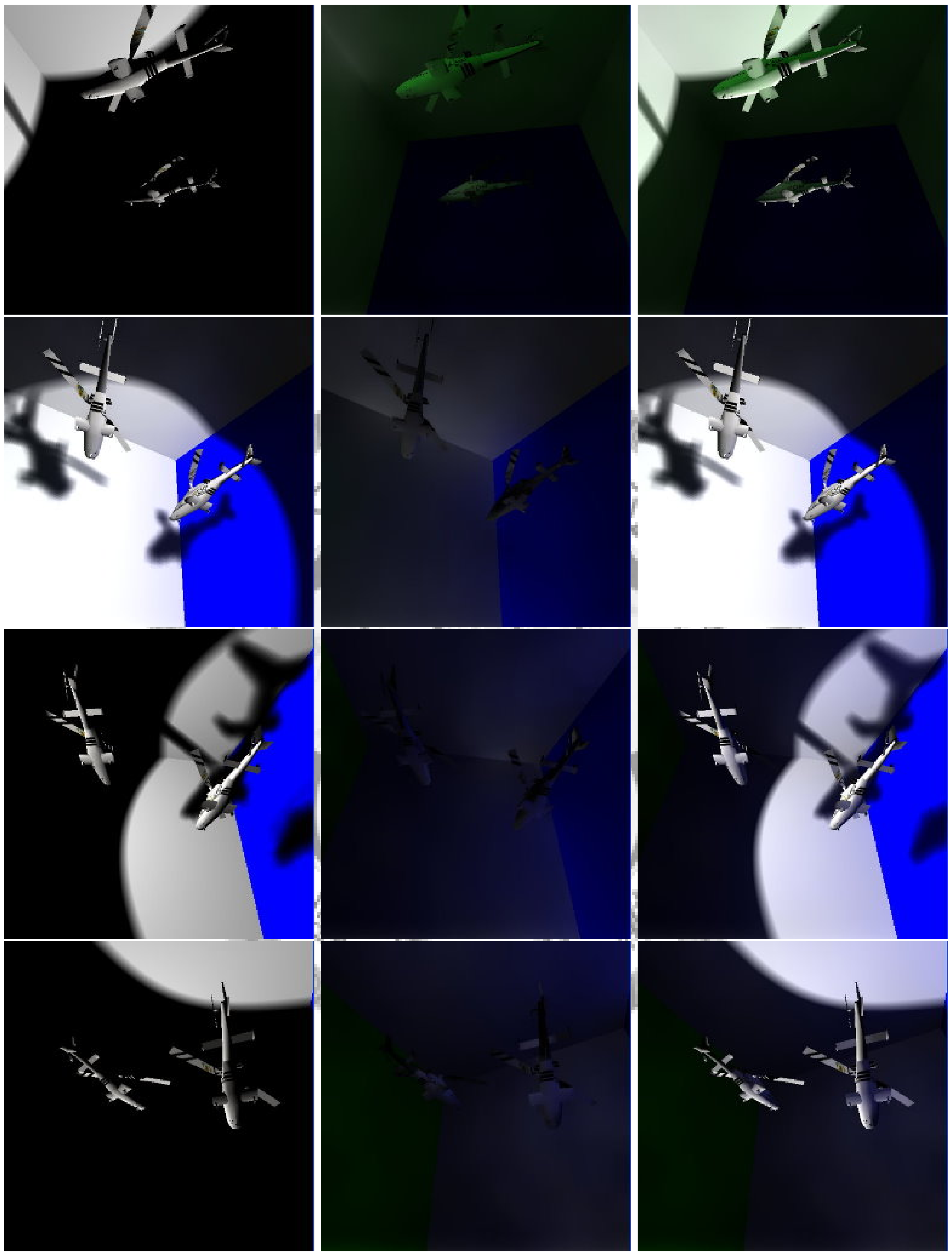


Figure 6.6: A series of images showing that the scene, lighting, and the camera are all changing. The left column shows the direct lighting only. The middle column shows the indirect lighting only. The right column shows the global illumination effects which is the combination of the images on the left column and middle column.

Model	Triangles	Samples per pixel	Resolution of IL map	Fps
Helicopters	1522	32	$512 * 512$	11.98
			$256 * 256$	12.04
			$128 * 128$	12.05
Helicopters	1522	64	$512 * 512$	8.16
			$256 * 256$	8.53
			$128 * 128$	8.54
Helicopters	1522	96	$512 * 512$	6.00
			$256 * 256$	6.39
			$128 * 128$	6.40
Helicopters	1522	128	$512 * 512$	4.76
			$256 * 256$	4.97
			$128 * 128$	5.01
Helicopters	1522	256	$512 * 512$	2.63
			$256 * 256$	2.72
			$128 * 128$	2.72
Helicopters	1522	512	$512 * 512$	1.40
			$256 * 256$	1.43
			$128 * 128$	1.43

Table 6.1: Comparison of different parameter settings, where the IL map means the indirect illumination map

Model	Triangles	Samples per pixel	Resolution of IL map	Fps
Venus and helicopters	3284	32	$512 * 512$	8.08
			$256 * 256$	8.45
			$128 * 128$	8.53
Venus and helicopters	3284	64	$512 * 512$	4.61
			$256 * 256$	4.88
			$128 * 128$	4.90
Venus and helicopters	3284	96	$512 * 512$	3.34
			$256 * 256$	3.47
			$128 * 128$	3.42
Venus and helicopters	3284	128	$512 * 512$	2.61
			$256 * 256$	2.63
			$128 * 128$	2.63
Venus and helicopters	3284	256	$512 * 512$	1.37
			$256 * 256$	1.39
			$128 * 128$	1.40
Venus and helicopters	3284	512	$512 * 512$	0.70
			$256 * 256$	0.71
			$128 * 128$	0.71

Table 6.2: Comparison of different parameter settings, where the IL map means the indirect illumination map

Chapter 7

Conclusion and Future Work

We presented a method to compute interactive global illumination in fully dynamic scenes and the light, the geometry, and the material are all changeable without any precomputation and assumptions. Our contribution is that we utilize the occlusion map and think it as the indirect light source map. And we accelerate the computation of the occlusion map by down-scaling. Thus the rendering process can be repeatedly computed every frame.

Although our method can be applied to the object with different kinds of materials, except for transparent objects. Basically, the occlusion map can be also used to find out the indirect light sources of the transparent objects. However, the intensities of those light sources refer to the direct illumination map and the generation of the direct illumination map with transparent objects needs some extra efforts. We will try to solve this problem by finding out an efficient way to generate the direct illumination effects with transparent objects.

Bibliography

[1] E. H. Adelson and A. P. Pentland. *The perception of shading and reflectance*. Cambridge University Press, New York, NY, USA, 1996.

[2] A. Blake and H. H. Bülthoff. Does the brain know the physics of specular reflection? *Nature*, 343, no. 6254:165–168, 1990.

[3] C. Dachsbacher and M. Stamminger. Splatting indirect illumination. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 93–100, New York, NY, USA, 2006. ACM.

[4] C. Dachsbacher, M. Stamminger, G. Drettakis, and F. Durand. Implicit visibility and antiradiance for interactive global illumination. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 61, New York, NY, USA, 2007. ACM.

[5] R. Dimitrov, L. Bavoil, and M. Sainz. Horizon-split ambient occlusion. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 1–1, New York, NY, USA, 2008. ACM.

[6] W. Donnelly and A. Lauritzen. Variance shadow maps. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165, New York, NY, USA, 2006. ACM.

[7] I. R. Fleming, W. Fleming, R. O. Dror, and E. H. Adelson. How do humans determine reflectance properties under unknown. In *Proceedings of CVPR Workshop on Identifying Objects Across Variations in Lighting: Psychophysics and Computation*, pages 347–368, 2001.

[8] R. W. Fleming, R. O. Dror, and E. H. Adelson. Real-world illumination and the perception of surface reflectance properties. *Journal of Vision*, 3:347–368, 2003.

[9] R. W. Fleming, A. Torralba, and E. H. Adelson. Specular reflections and the perception of shape. *J. Vis.*, 4(9):798–820, 9 2004.

[10] M. Hašan, F. Pellacini, and K. Bala. Matrix row-column sampling for the many-light problem. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 26, New York, NY, USA, 2007. ACM.

[11] R. Hunter and R. Harold. *The measurement of Appearance (2nd edition).* Wiley, New York, 1987.

[12] O. Kozlowski and J. Kautz. Is accurate occlusion of glossy reflections necessary? In *APGV '07: Proceedings of the 4th symposium on Applied perception in graphics and visualization*, pages 91–98, New York, NY, USA, 2007. ACM.

[13] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 376–381, New York, NY, USA, 2003. ACM.

[14] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–8, New York, NY, USA, 2008. ACM.

[15] P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on gpus. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 73–80, New York, NY, USA, 2007. ACM.

[16] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, New York, NY, USA, 2002. ACM.

[17] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: a scalable approach to illumination. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1098–1107, New York, NY, USA, 2005. ACM.

[18] R. Wang, J. Tran, and D. Luebke. All-frequency relighting of non-diffuse objects using separable brdf approximation. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 345–354, June 2004.

[19] L. Williams. Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274, New York, NY, USA, 1978. ACM.