Supplementary Note - Seamless Mipmap Filtering for Dual Paraboloid Maps

Zhenni Wang*, Tze Yui Ho†, Chi Sing Leung‡, and Eric. W. M. Wong*

1City University of Hong Kong, Hong Kong SAR, China

1. Introduction

This supplementary note documents some additional information we prepared for the paper Seamless Mipmap Filtering for Dual Paraboloid Maps. This information, though did not get the chance to be presented in the paper, might still be helpful for readers.

The soft shadow algorithm that we used is the omnidirectional version of CSS [XLH*13]. Evolving from the very first shadow mapping anti-aliasing algorithm (PCF) [RSC87] to the omnidirectional version of CSS, these algorithms deserve a detailed presentation. Just in cases readers might be interested, a comprehensive review about the soft shadow algorithms is provided in Section 4.

2. Frame rate

To demonstrate the speed advantage of dual paraboloid mapping, we measure the frame rate of dual paraboloid map (DPM) mipmaps and cubemap (CM) mipmaps using the omnidirectional soft shadow generation. The OpenGL Time Query mechanism allows us to communicate with the GPU directly and query timestamps with nanoseconds precision (see the OpenGL 4.6 specification, Section 4.3). Using the queried timestamps, we individually measure the processing time of the map generation, the mipmap generation and the shading.

The measured processing time of a series of scenes are reported in the performance table, Table 1, in millisecond. The series of scenes contain a sequence of a simple scene (each with a different tessellation level), two well-known scenes Sponza and San Miguel, a dragon scene and a plant scene.

2.1. The processing time of the map generation

We refer to the process of rendering the dynamic content to a DPM or a CM as map generation. As shown in Table 1, its processing time increases when the scene has more triangles, naturally.

If we look into the figures carefully, we can find that the processing time of DPM is smaller than that of CM by proportion. This behavior can be better understood in terms of the processing time ratio of DPM against CM. As shown in Table 1, starting from the ratio being almost 1:1, the ratio decreases gradually along with the number of the triangles increases and reaches 1:3 eventually. To be specific, the ratio of the Cube00 scene begins with 0.18 ms vs. 0.18 ms and ends with the San Miguel scene (low polygons) 29.50 ms vs. 89.50 ms. This behavior is consistent with the proposition in the paper that rendering to a dual paraboloid map is 3 times as fast as rendering to a cubemap theoretically.

2.2. The processing time of the mipmap generation

The next logical part of the algorithm is mipmap generation. As demonstrated in Table 1, the processing time of DPM mipmap generation keeps being faster than that of CM regardless of the number of triangles, and the processing time ratio of DPM against CM is roughly 1:2.6.

2.3. The processing time of the shading

For the final part of the algorithm, the shading, it includes looking up values from the DPM/CM mipmaps, calculating the shadow factors and rendering the scene. The shading processing time of DPM is similar to that of CM and makes the overall speed advantage of DPM less noticeable.

2.4. Overall performance

Now, we examine the overall performance of DPM and CM. The overall frame rate of DPM is consistently higher than that of CM.
Theoretically, rendering to a dual paraboloid map is 3 times as fast as rendering to a cubemap. Practically, how much faster depends on the three components mentioned above. Fig. 1 is the curve of the overall frame rate of DPM divided by the overall frame rate of CM, or the speed factor curve. We can evaluate the performance of DPM against CM intuitively by examining Fig.1.

When the number of triangles is small, the overall frame rate is dominated by the mipmap generation. Since the processing time of the mipmap generation does not depend on the number of triangles, the speed factor is basically a constant, around 2.2.

As the number of triangles increases, the overall frame rate will be dominated by the shading. This component, the shading, exerts the same amount of computational burden to DPM and CM and dilutes the DPM speed advantage. Numerically, the speed factor is being dragged down to around 1.6 - 1.9.

Then, along with the further increase of the number of triangles, the overall frame rate will be dominated by the map generation. The overall speed advantage of DPM then becomes more noticeable and causes the speed factor approaching its theoretical upper bound, 3.

3. Tessellation
3.1. The subdivision

Rendering to a DPM requires a reasonably tessellated scene. To facilitate the discussion, we choose a simple scene (see Fig. 2a), which can be described using a small number of triangles (42 triangles). Then, we apply subdivision to the scene and tessellate the scene with increasingly more triangles (60 - 1572900 triangles). The subdivided scenes are shown in Fig. 2b - Fig. 2j. Given the initial scene and its 9 subdivided, we have 10 scenes.

Table 1: Performance table. Map gen. mip gen. and shading are reported in ms, and total/frame-rate is reported in ms/fps.
3.2. The visual quality of DPM

Given these ten scenes, we measure the performance of DPM and CM (the visual quality and the frame rate) with each scene individually. The rendering images are shown in Fig. 3, and the frame rate is shown in Table 1.

The tessellation quality of a scene will influence the rendering quality of DPM. When the scene is tessellated with not enough number of triangles, the rendering result of DPM will have problem (see Fig. 3b). Raising the tessellation quality of the scene, or describing the scene with more triangles, the DPM rendering problem is alleviated correspondingly (see Fig. 3b - Fig. 3e). When the number of triangles reaches 420, the DPM rendering problem can no longer be visually perceived (see Fig. 3e). Compared to the initial scene (42 triangles) using CM (see Fig. 3a), the subdivided scene of 420 triangles using DPM provides a visually equivalent result (see Fig. 3e). Further increase the number of triangles, no further visual improvement is observed (see Fig. 3f - Fig. 3k). Measuring the MSE between all consecutive pairs of DPM rendering images, we have Fig. 4, which also suggests that no further visual improvement after subdividing the scene with 420 triangles.

3.3. The break-even point

CM is slower while DPM requires higher tessellation. It could be difficult to identify which factor is more important. To facilitate the discussion, we use the performance information reported Table 1.
to plot the frame rate graph in Fig. 5. The frame rate in the graph belongs to the scene Cube00 and its subdivided scenes. Shown in the figure, there are two curves with an intersection, which is the break-even point.

The break-even point suggests that tessellating the initial scene with 160k triangles stresses DPM to have the same frame rate as CM. As mentioned in Section 3.2, the subdivided scene of 420 triangles using DPM is visually equivalent to the initial scene using CM. 420 is far smaller a value w.r.t. 160k. This margin allows us to, without hurting the DPM speed advantage, re-tessellate scenes with comfortably more triangles to alleviate the problems of DPM.

3.4. Two well known scenes

Mentioned in Section 3.2, the scene Cube00 has no further visual improvement after being re-tessellated with 420 triangles. Using this as a reference, we roughly estimate that the scene is sufficiently tessellated if the edge length of all triangles is smaller than one-eighth of the scene dimension. To the scenes with higher geometric complexity, for instances Sponza and San Miguel, the triangles are not necessarily all sufficiently tessellated.

To enforce all triangles in the scene being sufficiently tessellated, we write a routine to re-tessellate triangles with edge length longer than a specific threshold. The re-tessellation continues until the edge length of all triangles is smaller than the threshold. Then, we use this routine to re-tessellate the two scenes, i.e. Sponza and San Miguel, to one-eighth of their scene dimension. Table 2 is the number of triangles before and after the re-tessellation of these scenes. As shown in the table, the triangles needed re-tessellation, so as the new triangles from the re-tessellation, is just a small proportion w.r.t. the triangles in the initial scenes. Fig. 6 shows the grid figures of the scenes before/after the re-tessellation.

Now, we examine the influence of the re-tessellation. Using the scene Sponza as an example, the rendering result of DPM does have problems (see Fig. 7b). With the re-tessellated scene, the DPM rendering problem can no longer be visually perceived (see Fig. 7c). Comparing DPM with the re-tessellated scene (see Fig. 7c) to CM with the initial scene (see Fig. 7a), we can also see that re-tessellating scenes can alleviate the DPM rendering problem and
Table 2: The initial/final triangles of Plant, Sponza and San Miguel

<table>
<thead>
<tr>
<th>Scene</th>
<th>Initial number of triangles</th>
<th>Triangles needed to be re-tessellated</th>
<th>Triangles increased</th>
<th>Final number of triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant</td>
<td>182520</td>
<td>0</td>
<td>0</td>
<td>182520</td>
</tr>
<tr>
<td>Sponza</td>
<td>262267</td>
<td>4755</td>
<td>16194</td>
<td>278461</td>
</tr>
<tr>
<td>San Miguel (low polygons)</td>
<td>5617451</td>
<td>585</td>
<td>2946</td>
<td>5620397</td>
</tr>
<tr>
<td>San Miguel</td>
<td>9980699</td>
<td>626</td>
<td>3669</td>
<td>9984368</td>
</tr>
</tbody>
</table>

makes DPM visually equivalent to CM. On top of the visual comparison, shown in Table 1, even with the additional amount of triangles from the re-tessellation, the speed advantage of DPM is still remarkable, which is 1.7 times as fast as CM. In fact, the impact of the re-tessellation to rendering speed is minimal. For example, the frame rate of DPM before and after the re-tessellation are 442 fps vs. 441 fps. In other words, using more triangles to obtain the speed advantage is worthwhile for DPM.

4. Shadow algorithms

4.1. Shadow mapping

![Shadow mapping](image)

Shadow mapping is a classical shadow generation algorithm. It is a two pass algorithm. The first one is to render a depth map (or a shadow map) from the light source perspective (see Fig. 8b). The second one is to generate shadows by comparing the distance from the light source to the surface with the corresponding depth in the shadow map (see Fig. 8a). The overall rendering result is shown in Fig. 8c.

One of the major problems of the shadow mapping algorithm is aliasing. Depending on the geometry (the light position, the eye position and the scene geometry), jaggy shadow can happen if the shadow map resolution is not higher enough (see Fig. 9a).

4.2. Percentage Close Filter (PCF) and anti-aliasing

Speaking of aliasing problem, the immediate solution would be anti-aliasing, i.e. filtering the results. For graphics application, the natural filtering tool would be mipmapping. However, directly filtering a shadow map cannot reduce the aliasing problem.

In order to blur the shadows, the filtering should happen after the
depth comparison (see Fig. 10b). The algorithm for this is called Percentage Close Filter (PCF) \cite{RSC87}. Fig. 11b is an anti-aliased shadow mapping result. Compared with Fig. 11a, the jaggy edge is blurred, or we can say the aliasing problem is reduced. Unfortunately, because of the depth comparison has to happen before the filtering, we can only use brute force filtering to implement PCF. In other words, we cannot use prefiltering techniques, like mipmapping, to accelerate PCF.

4.3. PCF and soft shadows

Although PCF is originally designed for anti-aliasing shadow mapping, its application is not necessarily limited to anti-aliasing. In fact, PCF is just a filtering tool for shadow mapping, i.e. blurring the shadows.

Considering the geometric assumption for the penumbra size estimation (see Fig. 12), \cite{Fer05} proposes a soft shadow algorithm based on PCF, namely Percentage-Closer Soft Shadows (PCSS).

4.4. Convolution Shadow Maps (CSM)

As mentioned in Section 4.2, the major drawback of PCF is that it can only be implemented using brute force filtering. When the filter size is relatively large, the rendering speed can become impractically slow. To handle this problem, a number of algorithms have been proposed, such as Exponential Shadow Maps \cite{AMS08}, Variance Shadow Maps \cite{DL06} and Convolution Shadow Maps (CSM) \cite{AMB07}. Roughly speaking, these algorithms use some mathematical tools to account for both filtering and comparison, such that the intermediate maps can be pre-filtered using mipmapping.

\cite{AMB07} uses CSM basis to decompose the depth values (i.e. the shadow map, e.g. Fig. 14a) in the frequency domain and obtains a series of basis textures (see Fig. 14b). CSM basis might sound
like something new. However, it is just a straightforward application of Fourier Series. Then, in the frequency domain, the filtering operation and the comparison operation are re-constructed simultaneously. In this case, the intermediate maps (see Fig. 14b) can be pre-filtered using mipmapping.

Without the major drawback of PCF, CSM provides us an efficient shadow mapping anti-aliasing tool even if the filter size is relatively large. Similar to PCF, CSM can also be extended for generating soft shadow. Convolution Soft Shadows (CSS) [ADM∗08] extends CSM for generating soft shadows. It applies Fourier Series approximation twice (i.e. CSM and CSM-Z) for the average blocker depth and the soft shadow factors. Analogous to CSM against PCF, CSS provides us an efficient soft shadow algorithm even if the light size is relatively large, which is impractical for PCSS. Two rendering results of CSM and CSS are shown in Fig. 15.

4.5. Omnidirectional soft shadows

Both PCSS and CSS are considering a planar light source. If we want to extend these two algorithms to the omnidirectional scenario, we need to modify the geometric assumption for the penumbra size estimation mentioned in Section 4.3. [XLH∗13] proposes an omnidirectional soft shadow algorithm based on CSS and another geometric assumption, namely Concentric Spherical Representation (see Fig. 16a), for the omnidirectional scenario.

Compared with CSS, one of the major differences of [XLH∗13] is that it uses the environment mapping tools (e.g. CM or DPM) to represent the basis textures. A rendering result of [XLH∗13] is shown in Fig. 16b.

References

[ADM∗08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. ACM transactions on graphics (TOG) 27, 3 (2008), 34. 7


[FER05] FERNANDO R.: Percentage-closer soft shadows. In SIGGRAPH Sketches (2005), p. 35. 6, 7


