Adaptive Spectral Mapping for Real-Time Dispersive Refraction

Damon Blanchette and Emmanuel Agu

Worcester Polytechnic Institute

Abstract. Spectral rendering, or image synthesis utilizing the constituent wavelengths of white light, enables the rendering of iridescent colors caused by phenomena such as dispersion, diffraction, interference and scattering. Dispersion creates the rainbow of colors when white light shines through a prism. Caustics, the focusing and de-focusing of light through a refractive medium, can be interpreted as a special case of dispersion where all the wavelengths travel along the same paths. In this paper we extend Adaptive Caustic Mapping (ACM), a previously proposed caustics mapping algorithm, to handle physically-based dispersion. Our proposed method runs in screen-space, and is fast enough to display plausible dispersion phenomena at real-time frame rates.

1 Introduction

This paper focuses on rendering physically accurate dispersive refraction at real-time frame rates. Figure 1 shows four examples of dispersive refraction, i.e. different wavelengths of light refracting at different angles, rendered with our technique.

The key difference between dispersion and caustics is that while the wavelengths of light are refracted along different paths in dispersion, all wavelengths are refracted along the same paths to generate caustics. By drawing on these similarities, we have extended a real-time caustics algorithm to render dispersive refraction in real time.

Specifically, we extended the Adaptive Caustic Mapping (ACM) algorithm [20] to perform real-time spectral dispersion. ACM is a real-time image-space method of generating refractive caustics on programmable graphics hardware. Our method, which we call Adaptive Spectral Mapping (ASM), begins with the ACM algorithm

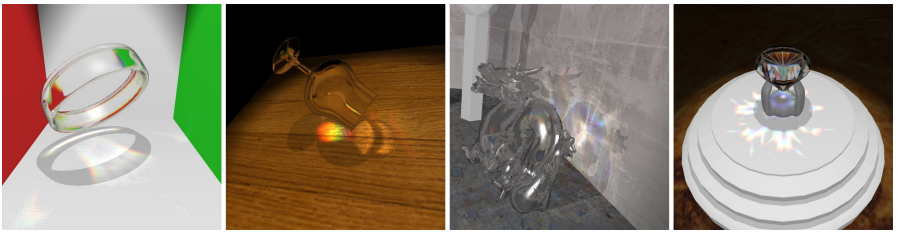

Fig. 1. Four images generated with our algorithm. All four scenes are using seven wavelength samples, and perform between 15 and 30 frames per second.

G. Bebis et al. (Eds.): ISVC 2012, Part I, LNCS 7431, pp. 81–91, 2012.

but adds spectral refraction calculations at object surfaces. To create spectral maps, we simulate external dispersion by refracting seven wavelengths at the surface of refractive objects. In a separate deferred rendering pass, we also calculate internal dispersion, which occurs when white light is split into component colors inside a refractive object such as the colors seen inside diamonds.

The rest of the paper is as follows. Section 2 describes related work. Section 3 gives some background on caustics rendering. Section 4 describes our technique for rendering spectral dispersion using Adaptive Spectral Maps (ASMs). Section 5 describes our implementation. Section 6 describes our results and section 7 is our conclusion and future work.

2 Related Work

Initially, spectral rendering was described in the context of ray or path tracing. Cook and Torrance [2] presented a method for rendering materials that takes into account light wavelengths and spectral energy distribution. Thomas [17] and Musgrave [13] presented specifically on dispersion using ray tracing methods.

Most recent real-time spectral rendering research uses the GPU to perform wavelength calculations. Guy and Soler [7] presented on the real-time rendering of dispersion inside gemstones. Kanamori et al. [12] published on the physically accurate display of rainbows under different atmospheric conditions. Ďurikovič et al. [3] presented an entire spectrally-based framework for interactive image synthesis that could display multilayered thin-film interference.

The work most similar to ours is by Sikachev et al. [16], in which they present spectral dispersion through gems. They differ in that their algorithm can only project dispersion onto planes as opposed to arbitrary surfaces as this paper presents.

3 Background

3.1 Caustics Rendering

Since our proposed technique is an extension of Adaptive Caustic Mapping, we now review the literature on caustics rendering. Kajiya [11] and Shirley [15] both described caustics generation using ray and path tracing algorithms. Jensen's photon mapping algorithm [10] can also generate caustic effects, using a special “caustics photon map” where extra photons are sent in order to create higher resolution data.

The idea for “caustic mapping,” a far better performance solution in which a special texture is created containing caustic data that is projected onto a scene similar to shadow mapping, began with the Shah et al. image-space technique [14]. Wyman et al. [18] presented a caustic mapping algorithm similar to Shah's that also operated in image-space. He then extended his own algorithm with a hierarchical caustics generation method [19] that used mipmaps and a reduced resolution version of the scene to increase algorithm speed. Wyman et al. later improved this hierarchical method to

yield Adaptive Caustic Mapping [20], which is described in detail in the next section since this work extends it.

3.2 Adaptive Caustic Mapping

We chose to extend Adaptive Caustic Mapping in particular because it solved several issues inherent in other caustic mapping algorithms: notably aliasing due to insufficient sampling and excessive temporal noise due to sampling variations. ACM uses an importance-based adaptive photon sampling algorithm that increases quality while also speeding up the rendering of caustics beyond other methods of similar quality, and in addition they utilize a deferred rendering process that displays refractive objects more quickly than other methods.

For a thorough description of ACM, we refer the reader to Wyman's original paper [20]. However, to aid in understanding, we will summarize the important points of the algorithm here and then describe our spectral dispersion extension.

ACM differs from other caustic mapping algorithms in its photon emission and refractive object "locating" phase. ACM starts with a reduced resolution view of the scene from a light source using mipmaps, and emits only a few regularly spaced photons into that image. In a loop, moving up one mipmap level at a time, each photon that actually hits a refractive object is subdivided into four new photons, increasing photon density and thus the resolution of the caustics. Photons that do not hit a refractive object are simply discarded and never processed. When this phase is completed, the photon buffer contains a high-resolution set of points that all intersect the surface of the refractive object. The photons are then refracted through the object using the normal values at each pixel and splatted onto the spectral map.

The display of the refractive objects in the scene is completed in a separate deferred pass at the end. Pixels that lie on a refractive object's surface are treated as photons, as with the caustics calculations. The photon is refracted once using the front-facing normal at the current pixel's location, and then a second time at the back-facing surface. It is then projected out to the background geometry, where a texture fetch is performed to get the color for that pixel.

4 Spectral Dispersion Using Adaptive Spectral Maps

To create our spectral maps, extensions were made to both the caustic generation algorithm and the deferred refraction algorithm. We chose to sample seven wavelengths that are evenly distributed through the visible spectrum.

Refraction angles of light between two mediums with different refractive indices can be calculated using Snell's Law [8]. Taking into account wavelength, the refractive index can be calculated using Cauchy's equation [1].

For our purposes, it is sufficient to use the following two-term form of Cauchy's equation, with λ representing wavelength, initially used by Musgrave [13]:

$$n(\lambda) = A + \frac{B}{\lambda^2} \ . \tag{1}$$

The A and B coefficients are based on physical measurements and can be found in tables in various sources such as physics textbooks and the Internet [5].

Just before each photon refracts at the front surface of the object, it is split into seven separate photons, and each new one is refracted according to the index of refraction generated by Cauchy's equation. Each of the seven photons is then refracted a second time on the back-facing surface of the object, after which its final position is calculated for splatting into the spectral map. Figure 2 illustrates this process.

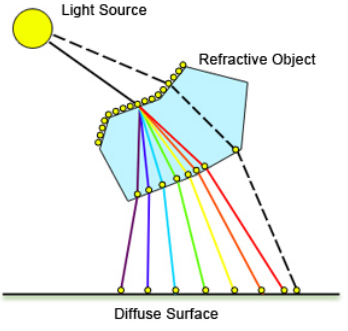

Fig. 2. Refraction using seven samples. The dotted line indicates the original ACM algorithm, and the solid lines are our extension.

In Figure 2, the small yellow circles are the photons. Until they are splatted into the spectral map, they are regarded only as wavelengths. Only just before splatting are they converted from a wavelength to an RGB value.

At this point the spectral map is complete and ready to be projected into the scene. The spectral map is a texture that contains the final locations of photons that have been refracted through the specular object according to their wavelengths and converted to RGB colors.

Once the spectral map has been created and projected into the scene like a shadow map, dispersion within the transparent object is calculated in a completely separate pass at the end. In ACM the color of each pixel on the surface of the refractive object is calculated using a single background color texture fetch. However, with ASM, we perform seven texture fetches – one for each wavelength sample. The location of the texture fetch on the background is calculated by following a ray of light as it passes through the refractive object and intersects the background texture.

The color of the texel chosen from the background texture is altered by the color of the wavelength that hits it, so if all wavelengths arrive at the same location or the same color, then the final color of the pixel on the refractive object is exactly the same as the background. The wavelength is converted to a RGB value here, when it is calculated based on the color of the background texture.

4.1 Filling the Gaps

One of the major issues with spectral rendering using discrete sampling of the spectrum is that gaps or empty portions occur in the resulting spectral map, as shown in figure 3.

We experimented with the simplest brute-force method of fixing gaps by testing our algorithm with 21 samples instead of seven. This indeed reduced the problem to a degree, but gaps still showed up where dispersion between colors was large. In addition, there was a 75% drop in frame rate with 21 samples.

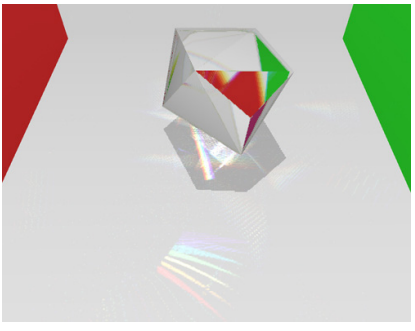

Fig. 3. Problems with discontinuous caustics when using seven samples. Each sample color is clearly visible, with gaps between the colors.

Sikachev et al. proposed interpolating colors between the caustics that do exist in order to solve the color gap problem [16]. They integrate the interpolation results for each point by performing additive blending, and use a given step size which is taken in the view space coordinates.

We propose a similar method using texel marching: for each texel in the caustic map that is not already illuminated, a step is performed one pixel at a time, horizontally and vertically from it. If a colored texel is found, its color is mixed with the current texel's color, akin to interpolating the colors around a gap to fill it in.

Figure 4 illustrates the idea behind our gap-filling algorithm.

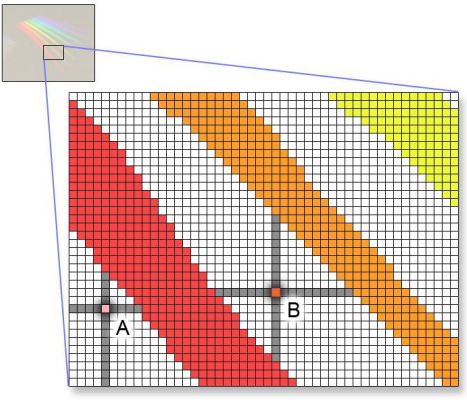

Fig. 4. A diagram showing how our filling algorithm works. Each grid square is one texel in the spectral map.

The gray pixels emanating horizontally and vertically from A and B represent the texel marching step. Texel A is set to a light red color because its vertical and horizontal neighbors are either from the red band or from nothing at all. Pixel B is set to a mixture of the red and orange color bands due to its proximity to both colors in the spectral map. Figure 5 shows images of the spectral map before and after our gap-filling procedure.

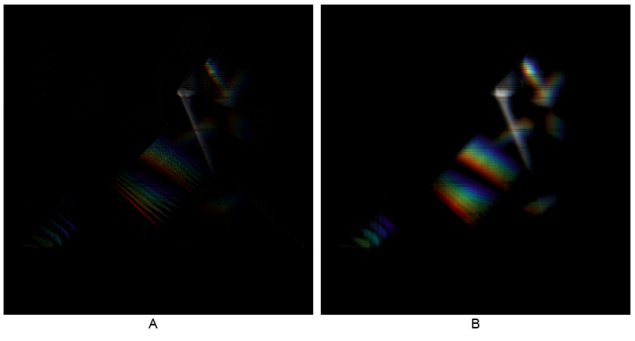

Fig. 5. The spectral map texture before (A) and after (B) filling in the gaps

The chosen number of steps is a tradeoff between better filling with blurry dispersion or sharper dispersed colors with the possibility of gaps not completely filled in. We found that a step size of 20 is a good balance between gap filling and blurriness.

5 Implementation

We implemented using C/C++ and OpenGL 4.2, with vertex, geometry, and fragment shaders written in GLSL. The video card utilized was an NVIDIA GeForce GTX480 in a Windows 7 environment.

We began by implementing Wyman's Adaptive Caustic Mapping algorithm [20].

The photon splatting shaders were extensively modified by the insertion of a new geometry shader to perform photon splitting into seven samples and to handle refraction for each wavelength. The fragment shader was altered to convert the wavelength values to RGB.

Figure 6 shows the entire pipeline for this project from beginning to final image, and each box describes a separate render pass. The yellow, blue, and green boxes in the background show how those passes are being rendered – whether it is from the light's view, from the camera's view, or in image space (on a full-screen quad). This diagram also compares our ASM algorithm with the original ACMs: each white box is unaltered from the original ACM algorithm, light red boxes are altered from the original ACMs, and pass 5, the dark red box, is a completely new pass.

Pass 4's alterations are shown in the pseudocode in figure 7. Both parts 2 and 3 were edited to use our Cauchy equation-calculated refractive indices per wavelength.

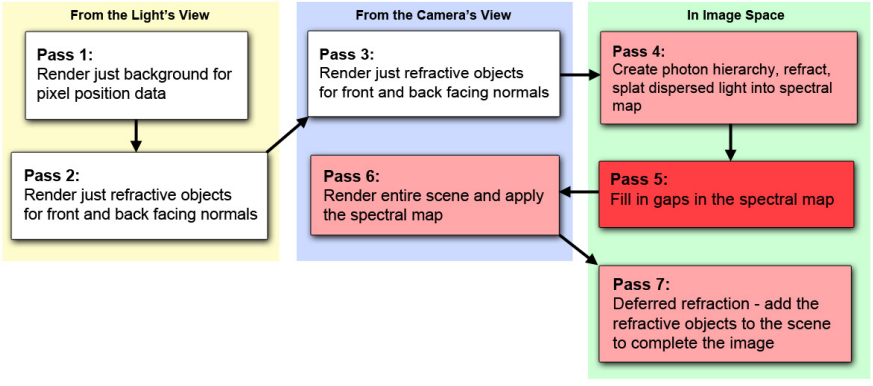

Fig. 6. Our rendering pipeline from beginning to end, completed each frame. This diagram shows both our ASM algorithm and the original ACM algorithm: the red boxes are passes altered from ACM, and pass 5, with the dark red background, is a completely new pass.

```
1. for each photon wavelength
2.     refract at front surface;
3.     refract at back surface;
4.     emit photon;
5.     compute gaussian intensity;
6.     convert wavelength to RGB;
7.     splat into spectral map;
```

Fig. 7. Photon splat pseudocode for pass 4 from figure 6

Part 4 is a geometry shader requirement, just there to emit the photon/vertex after refraction. Part 5 is identical to the ACM author's original code. In our newly created part 6, we convert the wavelength for that photon into an RGB value, described in the next paragraphs. Part 7 is a simple call to gl_FragData, required for all fragment shaders.

Since it is known exactly which wavelengths are being sent to the splat shader, a constant red, green, and blue value can be set for each one. As mentioned previously, these values must be carefully chosen to make sure they sum to white [13]. The color approximations are based on using the CIE color matching functions to get the relative contributions of light from wavelength, and converting them to XYZ color space coordinates [4]. The color matching functions can be described by the integral:

$$X = \int_0^\infty I(\lambda)\bar{x}(\lambda)d\lambda. \tag{2}$$

Where $I(\lambda)$ is the spectral power distribution, $\bar{x}$ is the color-matching function, and λ is the wavelength in nanometers. The Y and Z components are calculated in the same way. From the XYZ coordinates, it is possible to get RGB values using the CIE color space. In this way, a particular pixel's color in the scene is summed for each

wavelength-specific photon that hits it. If all wavelengths end up on the same pixel, it will be white. If only one photon wavelength hits a pixel, the pixel will only be that color.

After the spectral map is created, the next pass performs our gap filling shader to take care of gaps and any noise or missing pixels in the spectral map as described in section 4.1. Gap-filling is not performed on the surface of the refractive object as it is with the spectral caustic map because gaps and missing pixels do not occur there. This is due to the fact that photons are not being splatted into a separate map – colors are being pulled from background geometry, which always exists (or is black if nothing is there).

6 Results

Table 1 contains performance data for our algorithm using both seven and 21 samples compared to Adaptive Caustic Mapping. As can be seen in the table, the extra photons needed for dispersion reduces performance in some scenes, and increasing wavelength samples severely reduced speed in all scenes. The sphere, at least with seven samples, still performs at the same speed as with ACM, most likely due to its simplicity and the small size of its footprint from the view of the light. The gem, being composed of far fewer faces than all the other objects, still performs slower than the sphere since it is rendered onto more fragments due to its size, which is an image-space algorithm issue discussed in the following paragraphs. We believe the glass on the table also performed so poorly because of its size in the light's view.

Table 1. Frame rates for our test objects

Object	Faces	ACM	ASM, 7 samples	ASM, 21 samples
Sphere	5120	60	60	19
Ring	65536	27	20	9
Gem	24	60	40	10
Bunny	138902	12	10	6
Glass on Table	12137	60	16	5

Table 2. Table showing relative number of pixels taken up by a refractive sphere as seen from the light source and a frame rate comparison

Frame Rate (frames per second)	Percentage of total pixels covered by object
60	2.5% (sphere on "floor" of Cornell box)
40	4%
30	7%
20	13%
10	33%

Since this algorithm runs in image space, the number of pixels covered by the refractive object from the light's view has an impact on frame rate. The closer the object is to the light, the more pixels involved in caustic calculations, and the slower the performance. On the other hand, this results in better the quality. The relationship between this number of pixels and frame rate is closely tied - table 2 shows what happens as the sphere is moved closer to the light source.

Figure 8 shows a comparison between a screenshot of our software and a ground truth image, which is the same scene rendered with the offline engine LuxRender. Part A was performing at 40 frames per second, and part B took one hour to render. A few things to note: first is that the large caustic directly under the gem is very similar in shape, size, and color in both images. However, there are a couple discrepancies, one of which can easily be explained.

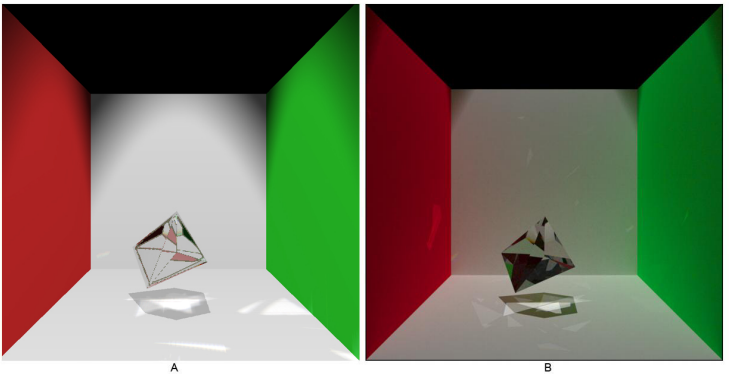

Fig. 8. A comparison between our algorithm, A, and a ground-truth render of the scene, B

The caustics on the walls in the ground truth image were created by reflective caustics, which our algorithm, and indeed ACM, does not simulate. A reason for this is that reflective caustics are more difficult to simulate due to possible extreme changes in a light path's direction, though this could be overcome by using a cube map for the spectral map [14]. The extra caustics on the floor of our screen capture that do not appear in the ground truth are possibly there as a result of imperfect sampling of the refractive object for photon placement.

As for the gem itself, looking closely near the top one can see there is a yellowish tinge above the red color, and on the right there is a green color at the intersection of the walls. These color shifts are a result of utilizing spectral calculations, and would not be present using an algorithm that does not account for the wave nature of light.

There are other dissimilarities between the images as well - the most obvious being the color of the gem to the bottom left. The refraction appears to be correct however, because the lines and positioning of the walls are quite similar.

7 Conclusion and Future Work

We have presented Adaptive Spectral Mapping, a spectral dispersion extension to the proposed algorithm Adaptive Caustic Mapping. Our algorithm displays a plausible

approximation of the dispersion phenomenon of light, and does so at interactive and real-time frame rates. Our ASM algorithm is one of the first of its kind, bringing spectral rendering one step closer to being fully displayed in real-time contexts such as games.

There are some limitations to our algorithm, however. The gap-filling procedure creates horizontal and vertical lines in some situations due to our sampling process. This could be ameliorated with a more random sampling method, which might introduce a temporal cohesion issue, but there would be fewer vertical and horizontal lines.

Many opportunities exist for future directions of research. The first could be to extend ASMs to simulate reflective caustics. Others include extending ASMs to display other spectral phenomena that require wavelength calculations, such as diffraction and thin-film interference. Integrating a fast volumetric caustics algorithm with ours would produce beautiful images, along the lines of recent research such as [9]. A distance-aware blurring algorithm such as Screen-Space Soft Shadows [6] could be modified to work with our spectral maps to make them more physically accurate, and might also help with more distant gaps as well.

References

1. Chartier, G.: Introduction to Optics. Springer (2005)
2. Cook, R., Torrance, K.: A Reflectance Model for Computer Graphics. ACM Transactions on Graphics 1(1), 7–24 (1982)
3. Ďurikovič, R., Kimura, R.: Spectrum-Based Rendering Using Programmable Graphics Hardware. In: SCCG 2005 Proceedings, pp. 233–236 (2005)
4. Evans, G.F., McCool, M.D.: Stratified Wavelength Clusters for Efficient Spectral Monte Carlo Rendering. In: Graphics Interface (1999)
5. Gooch, J.W.: Encyclopedic Dictionary of Polymers, vol. 1. Springer (2010)
6. Gumbau, J., Chover, M., Sbert, M.: Screen Space Soft Shadows. In: GPU Pro., pp. 477-490. AK Peters (2010)
7. Guy, S., Soler, C.: Graphics Gems Revisited: Fast and Physically-Based Rendering of Gemstones. ACM Transactions on Graphics 23(3), 231–238 (2004)
8. Hecht, E.: Optics, 4th edn. Addison Wesley (2001)
9. Hu, W., Dong, Z., Ihrke, I., Grosch, T., Yuan, G., Seidel, H.-P.: Interactive Volume Caustics in Single-Scattering Media. In: Proc. 2010 ACM SIGGRAPH Symp. Interactive 3D Graphics and Games, I3D 2010, pp. 109–117 (2010)
10. Jensen, H.W.: Global Illumination Using Photon Maps. In: Rendering Techniques 1996, pp. 21-30 (1996)
11. Kajiya, J.: The Rendering Equation. In: SIGGRAPH 1986 Proceedings, vol. 20(4), pp. 143–150 (1986)
12. Kanamori, S., Fujiwara, K., Yoshinobu, T., Raytchev, B., Tamaki, T., Kaneda, K.: Physically-Based Rendering of Rainbows Under Various Atmospheric Conditions. Computer Graphics and Applications (PG), 39–45 (2010)
13. Kenton Musgrave, F.: Prisms and Rainbows: A Dispersion Model for Computer Graphics. In: Proc. of Graphics Interface 1989, pp. 227–234 (1989)
14. Shah, M., Konttinen, J., Pattanaik, S.: Caustics Mapping: An Image-Space Technique for Real-Time Caustics. IEEE Transaction on Visualization and Computer Graphics (2005)

15. Shirley, P.: A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes. In: Proceedings on Graphics Interface, pp. 205–212 (1990)
16. Sikachev, P., Tisevich, I., Ignatenko, A.: Rendering Smooth Spectrum Caustics on Plane for Refractive Polyhedrons. In: 18th International Conference on Computer Graphics Graphicon 2008, pp. 172–176 (2008)
17. Thomas, S.: Dispersive Refraction in Ray Tracing. The Visual Computer 2(1), 3–8 (1986)
18. Wyman, C., Davis, S.: Interactive Image-Space Techniques for Approximating Caustics. In: ACM Symposium on Interactive 3D Graphics and Games, pp. 153–160 (2006)
19. Wyman, C.: Hierarchical Caustic Maps. In: ACM Symposium on Interactive 3D Graphics and Games, pp. 163–171 (2008)
20. Wyman, C., Nichols, G.: Adaptive Caustic Maps Using Deferred Shading. Computer Graphics Forum 28(2), 309–318 (2009)