Reflection Space Image Based Rendering

Brian Cabral, Marc Olano, Philip Nemec*
SGI

Abstract

High quality, physically accurate rendering at interactive rates has widespread application, but is a daunting task. We attempt to bridge the gap between high-quality offline and interactive rendering by using existing environment mapping hardware in combination with a novel Image Based Rendering (IBR) algorithm. The primary contribution lies in performing IBR in reflection space. This method can be applied to ordinary environment maps, but for more physically accurate rendering, we apply reflection space IBR to radiance environment maps. A radiance environment map pre-integrates a Bidirectional Reflection Distribution Function (BRDF) with a lighting environment. Using the reflection-space IBR algorithm on radiance environment maps allows interactive rendering of arbitrary objects with a large class of complex BRDFs in arbitrary lighting environments. The ultimate simplicity of the final algorithm suggests that it will be widely and immediately valuable given the ready availability of hardware assisted environment mapping.

CR categories and subject descriptors: I.3.3 [Computer Graphics]: Picture/Image generation; I.3.7 [Image Processing]: Enhancement.

Keywords: interactive rendering and shading, texture mapping, reflection mapping, image based rendering.

1 INTRODUCTION

Offline rendering algorithms have to a great extent conquered physically accurate photo-realism and complex synthetic shading. A result of over twenty years of research [1, 4, 5, 12, 15, 19, 21, 23], these techniques all solve the lighting or rendering equation [16] in some manner. The outstanding rendering challenge now becomes how to increase the performance of sophisticated shading algorithms without losing the advancements made in quality. Specifically, we are interested in interactive algorithms that change the nature and work-flow of artists, designers and scientists.

This implies that many orders of magnitude in performance improvements must be found. Traditionally, this has been accomplished by vastly simplifying the approximations used in the shading and lighting equations – resulting in a significant loss in complexity and quality. We take a modest, intermediate step toward interactive photo-realistic rendering – an algorithm that uses IBR techniques to approximate the integral lighting equation.

*e-mail: {cabral,olano,nemec}@sgi.com

Environment mapping is one method used to improve the realism of interactive rendering. As originally described by Newell and Blinn [1], a simple environment map is used to quickly find reflections of distant objects from a perfectly mirrored surface. Other researchers refined this notion by generalizing the BRDF used [3, 11, 19, 21], though some of these refinements lost the interactivity of simple environment mapping.

Another method used to bridge the gap between realism and interactivity is image based rendering [18]. IBR avoids solving the lighting equation during interactive rendering by warping existing photographs or images. These images can be thought of as radiance maps [8], and generalized to light fields [17] and lumigraphs [10]. This works well for predefined scenes or images, but not for dynamically changing synthetic objects.

Recently Debevec [5] combined captured environment maps and synthetic objects to produce compelling renderings with both synthetic objects and image based environments. His techniques do not work at interactive rates since he computes the lighting equation integration as he renders using RADIANCE [25].

We build upon the work of Debevec by pre-integrating the lighting equation to allow interactive rendering – in part by constraining the geometry of the global radiance environment maps. This introduces a view-dependence in the generated maps, which we overcome with a new form of IBR in reflection space. The result is interactive rendering of synthetic objects using a wide range of BRDFs in arbitrary lighting environments.

There are two primary contributions in this paper. The first is the application of IBR techniques in reflection space. We call this method reflection space IBR – even though we are not operating on normal images, nor is the result a rendered image. Second is our hybrid rendering algorithm. While we use IBR techniques, our algorithm and the nature of environment maps allows us to to use new BRDFs and new geometry not found in the original source images – thus extending the class of renderings possible with interactive IBR algorithms.

2 RADIANCE ENVIRONMENT MAPS

A traditional environment map records the incident radiance, L_i, from each direction. The two most common representations are the sphere map and cube map. A sphere map is a view-dependent representation, equivalent to an orthographic view of a reflective sphere. Note that it is not necessary to render an orthographic view when using a sphere map, for example the sphere mapping in OpenGL 1.2 includes the appropriate correction for perspective views. A cube map is a view-independent representation, created by projecting the environment onto the faces of a cube. Other representations are also possible, for example the parabolic map used by Heidrich and Seidel [13].

Instead of recording incoming radiance, a radiance environment map records the total reflected radiance, L_r, for each possible surface orientation. It is defined by the classic lighting equa-

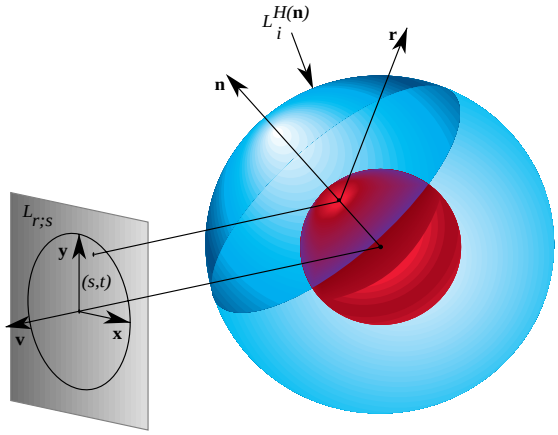

Figure 1: A radiance environment sphere map, $L_{r;s}$, is defined on the image plane shown on the left. Each point (s,t) in $L_{r;s}$ is associated with a normal, $\mathbf{n}$, and a reflection vector, $\mathbf{r}$. The normal specifies the hemisphere, $H(\mathbf{n})$ over which $L_i^{H(\mathbf{n})}$ is defined.

tion [14]:

$$L_r(\theta_r, \phi_r) = \int_H f_r(\theta_i, \phi_i; \theta_r, \phi_r) L_i(\theta_i, \phi_i) \cos\theta_i d\omega_i$$

Thus, L_r is the environment (L_i) modulated by the BRDF (f_r). H is the hemisphere above the surface, which varies with surface orientation (see figure 1). Each texel in a radiance environment map captures the pre-integrated value for L_r for one possible orientation of H. Since the radiance environment map is indexed by orientation, we can choose any of the storage representations used for traditional environment mapping. Figure 1 shows a sphere map representation. Heidrich and Seidel [13] use a similar technique of pre-integrating a BRDF and environment.

The pre-integrated radiance environment map introduces a couple of restrictions for rendering. Since all surface points that have a common normal use the same reflected radiance, only the lighting contribution from a distant environment can be captured, not reflections of local objects or inter-reflections of a single surface. Also, we are restricted to isotropic BRDFs; with only a single reflected radiance stored in the map per surface normal there is no way to record BRDF variation with rotation around the normal.

We also introduce an approximation to the true lighting equation. A radiance environment map is computed with a single view direction, so it is incorrect to use it with a perspective view, where the view direction changes from pixel to pixel. While graphics hardware corrects for perspective-induced changes in mirror reflection direction, this correction is not always appropriate for the radiance environment map. We render perspective views anyway and accept the (usually minor) resulting errors as part of the price for interactivity.

Obtaining Radiance Environment Maps

One method to obtain radiance environment maps is to take photographs in the desired environment of a physical sphere whose surface BRDF matches that of the target object. The photographs should be taken with a narrow field of view lens to approximate an orthographic projection and to minimize the reflection of the camera. The resulting images are the radiance environment maps, with the integration done by nature. Our algorithm requires several radiance environment maps, so we require several such images along with the camera orientation for each.

A second method is to compute the lighting integral numerically given a desired BRDF and lighting environment. The lighting environment should be known with high dynamic range for good integration results. Such environments can be captured through photographs by the methods of Debevec [6], or rendered with a package like RADIANCE [25]. We have used six photographs or images to represent L_i, arranged as a cube environment map [24]. Since the BRDF and the environment map, L_i, are decoupled the lighting environment can be reused to compute L_r for many different surface types. Results using maps computed in this way are shown in figure 3.

3 REFLECTION SPACE IBR

With conventional IBR, the light field is sampled by a discrete set of images. For our algorithm, these samples are a set of radiance environment maps taken from different viewpoints. These maps must be warped to match a new point of view, then blended together.

In addition to matching the viewpoint, the warping correlates features on the different maps. For traditional IBR, the image correlation may require only an affine or projective warp [22]. For general light fields it can require gathering light rays in a variety of discontinuous patterns [17].

Since each point in a radiance environment map is an integration of the environment and BRDF, the warp that best correlates features in the environment can vary from BRDF to BRDF. By choosing a warp that models the BRDF well, we can significantly reduce the number of radiance environment maps required for good reconstruction. If the BRDF is a perfect mirror and the warp models it as a perfect mirror, we need only sample well enough to catch the highest frequencies in the environment. If the warp does not match the BRDF, we must sample well enough for the product of the highest frequencies in the BRDF and environment. This is because the lighting integral is essentially a convolution of the BRDF with the environment.

For BRDFs that are principally reflective, we use a warp that matches the reflection directions of the different maps. So a point on the source image warps to the point on the destination image that reflects in the same direction. Primarily diffuse BRDFs suggest a warp that matches surface normal directions. We can find a well-matched warp for any BRDF that is radially symmetric about a principal direction and does not change shape across the surface. With such a BRDF, the same area of the environment will be integrated for corresponding points from different views.

Lambertian diffuse reflection and Phong specular reflection both satisfy this restriction, but most more realistic BRDFs do not. Fortunately, since the radiance environment maps sample a smooth, continuous function, we can effectively handle a much wider class of BRDFs that are close to the symmetric ideal without requiring a large number of sample maps. For example, we have used a numerically computed BRDF with Fresnel effects and diffuse, specular and backscatter components. For this BRDF, we use a warp that matches mirror reflections. It works because the specular lobe is the only high-frequency component of the BRDF and its shape does not vary too much from texel to texel. The Fresnel effects are naturally handled by the method and the other components do not require a large number of sample maps because they are low frequency.

Once the sample maps have been warped to the new viewpoint, they must be combined with some reconstruction filter. Oppenheim and Schafer [20] describe many sampling and reconstruction choices. For simplicity and efficiency, we use linear interpolation between neighboring images. The linear interpolation uses a spherical form of barycentric weights, presented in section 3.4. Thus, for any given viewpoint, the three nearby radiance environment maps are warped and blended to create an approximation to the new map (see figure 4).

3.1 Sampling View Directions

Certain environment map representations (e.g. cube maps) are viewpoint independent, while others (e.g. sphere maps) depend on the viewpoint. In contrast, each radiance environment map, whether it is stored in cube map, sphere map or another form, is correct for only a single viewpoint. This is because the radiance environment map captures Fresnel reflectance and other view-dependent effects.

As alluded to above, the view-dependence does limit the use of each map to only one view. This limitation is overcome by precomputing a set of maps – denoted $L_{r,j}, j \in \{0 \ldots N-1\}$ – at various viewpoints. The unit view vectors can be thought of as points lying on a sphere. We use reflection-space IBR to reconstruct the map for rendering from the $L_{r,j}$ maps, but we still require reasonable coverage of the sphere of possible view directions to avoid aliasing artifacts. We have used one L_r for each viewpoint defined at the vertices of an icosahedron. This number of samples has been sufficient for the environments and BRDF we have employed and is desirable because its symmetry means that each viewpoint is handled in an unbiased manner.

3.2 Map Warping

Each warp is between a source map, $L_{r;s}$ (from the precomputed set $L_{r,j}$) and a destination map, $L_{r;d}$ (for the current rendering viewpoint). Points in these maps will be called $\mathbf{p}_s$ and $\mathbf{p}_d$ respectively.

For each map point, $\mathbf{p}$, there is a vector $\mathbf{r}$ along the central reflection direction of the BRDF. For Phong specular or perfect mirror reflectors, $\mathbf{r}$ is the geometric reflection vector. For diffuse surfaces $\mathbf{r}$ is the surface normal. To assist in the warp, we define an invertible function

$$g : \mathbf{p} \rightarrow \mathbf{r}$$

$g(\mathbf{p})$ depends on both the BRDF and the map representation. It is most easily defined in terms of a local coordinate system for each map, so we also have a transformation per map to convert the local coordinate system to a common global space

$$T : \mathbf{r} \rightarrow \overline{\mathbf{r}}$$

The composition of these functions defines the full warp from $\mathbf{p}_s$ to $\mathbf{p}_d$:

$$\mathbf{p}_d = g_d^{-1} \circ T_d^{-1} \circ T_s \circ g_s(\mathbf{p}_s)$$

This takes a point in $L_{r;s}$ (defined by s and t texture coordinates for a sphere map representation). It is converted first to a 3D reflection vector in the local coordinate system associated with $L_{r;s}$. This 3D vector is transformed to the global space, then to a vector in the local coordinate system associated with $L_{r;d}$. Finally, the resulting vector is converted to a point in $L_{r;d}$ (once again given by two texture coordinates if we use the sphere map representation).

3.3 Specific Warps

We will derive two specific warp functions. Both use a sphere-map representation for $L_{r;s}$ and $L_{r;d}$. The first is for BRDFs where the central reflection direction is the surface normal. The second is for BRDFs where the central reflection direction aligns with the mirror reflection direction.

For both warps, the local coordinate system associated with each map is aligned with the camera used to create the map. The x-axis points right, the y-axis points up and the z-axis points from the origin toward the camera. Thus transformations T_s and T_d are defined as 3x3 matrices with columns equal to three axes expressed in global coordinates.

The surface normal warp uses g_{normal}:

$$g_{normal}(s,t) = \begin{pmatrix} 2s-1 \\ 2t-1 \\ \sqrt{1-(2s-1)^2-(2t-1)^2} \end{pmatrix}$$

$$g_{normal}^{-1}(x,y,z) = (x/2+.5, y/2+.5)$$

We base the mirror reflection warp, g_{mirror} on the x, y and z produced by g_{normal}:

$$g_{mirror}(s,t) = \begin{pmatrix} 2xz \\ 2yz \\ 2z^2-1 \end{pmatrix}$$

$$g_{mirror}^{-1}(x,y,z) = g_{normal}^{-1}\left(\frac{(x,y,z+1)}{\sqrt{x^2+y^2+(z+1)^2}}\right)$$

Since we have graphics hardware to do mirror reflections with a sphere map, we modify the final stage of both warps to use g_{mirror}^{-1}. The following functional composition chains define the two warps:

$$\mathbf{p}_d = g_{mirror}^{-1} \circ T_d^{-1} \circ T_s \circ g_{normal}(\mathbf{p}_s)$$
$$\mathbf{p}_d = g_{mirror}^{-1} \circ T_d^{-1} \circ T_s \circ g_{mirror}(\mathbf{p}_s)$$

Performing three of these warps per texel in the target map for every rendered frame is expensive and impractical for an interactive application. A fast, accurate approximation is possible by rendering the destination sphere map as a tessellated disk. Texture coordinates at each mesh vertex are chosen according to the warping function, and the source image is used as a texture while rendering the disk into the frame buffer. To account for the non-linearity of the warp functions, the mesh is finer toward the edges of the disk and coarser near the center. The 3D coordinate system associated with the destination map changes as the view moves, but the same disk tessellation can always be used. The reflection vectors, $\mathbf{r}_d$, also remain constant for each mesh vertex and can be precomputed.

The piecewise linear approximation to the warp is accurate for most of the sphere map area. Because we use a sphere map representation, the mirror warp has a singularity at the limit of extreme grazing reflections around the edge of the map – the reflection direction exactly opposite the view vector. The warp equation from $L_{r;s}$ to $L_{r;d}$ fails at this singularity.

We can locate $\mathbf{p}_d$ for the singularity by warping the problem reflection direction (the negated source map view vector) into the destination map. Near this point in the destination map, the source map will become pinched and unreliable instead of warping cleanly. We use a simple distance from $\mathbf{p}_d$ in the destination map to weight our confidence in the warped source image. This weight is used to fade the contribution of each source near its respective singularity.

The textured disk method accelerates the warping operation in two ways. First, s and t are not explicitly calculated for all the

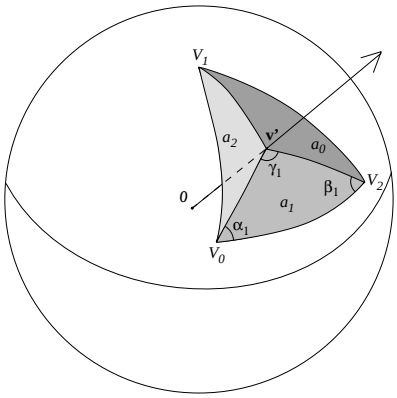

Figure 2: Illustrated is the spherical patch defined by $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$ associated with a particular point of view given by $\mathbf{v}_d$. By definition $\mathbf{v}_d$ defines in the line of sight of the viewer and in general forms three spherical triangles within the larger spherical triangle patch. Areas a_0, a_1 and a_2 represent the three weights for the sphere maps associated with vertices $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively, where α_1, β_1, and γ_1 are the dihedral angles used to compute a_1.

points on the sphere map, only at the vertices of the mesh. Second, a major bottleneck in performing the warp is accessing memory associated with the source and destination maps. We leverage the rendering and texturing hardware to solve this memory bottleneck.

3.4 Spherical Barycentric Interpolation

Once the warps have taken place the warped images must be blended. Our interpolation scheme is a spherical variant of classic affine barycentric interpolation, as defined in Farin [7]. Classic barycentric interpolation uses the ratio of the areas of triangles, we instead use the ratio of spherical triangles.

Any given view vector, $\mathbf{v}_d$, will in general lie within a spherical patch as illustrated in figure 2. Each vertex, $\mathbf{v}_i$ of this spherical triangle is the view vector for one of the source images that have been warped to match $\mathbf{v}_d$. $\mathbf{v}_d$ is used to form three interior spherical triangles. The weight for the source image at vertex i is a ratio of the areas of the spherical triangle opposite $\mathbf{v}_i$ and the overall spherical triangle. The area of an interior spherical triangle, a_i, on a unit sphere is given by the spherical excess formula [2]:

$$a_i = \alpha_i + \beta_i + \gamma_i - \pi; i \in 0, 1, 2$$

The dihedral angles α_i, β_i, and γ_i are defined as:

$$\begin{aligned}
\alpha_i &= \cos^{-1}\left((\mathbf{v}_d \otimes \mathbf{v}_{(i-1)\oslash 3}) \cdot (\mathbf{v}_{(i+1)\oslash 3} \otimes \mathbf{v}_{(i-1)\oslash 3})\right) \\
\beta_i &= \cos^{-1}\left((\mathbf{v}_{(i+1)\oslash 3} \otimes \mathbf{v}_d) \cdot (\mathbf{v}_{(i+1)\oslash 3} \otimes \mathbf{v}_{(i-1)\oslash 3})\right) \\
\gamma_i &= \cos^{-1}\left((\mathbf{v}_{(i+1)\oslash 3} \otimes \mathbf{v}_d) \cdot (\mathbf{v}_{(i-1)\oslash 3} \otimes \mathbf{v}_d)\right)
\end{aligned}$$

Where $\otimes$ is the normalized cross product and $\oslash$ is an index-wrapping operator, defined as

$$a \oslash b = \begin{cases} b-1 & \text{if } a < 0 \\ 0 & \text{if } a \geq b \\ a & \text{otherwise} \end{cases}$$

4 RENDERING ALGORITHM

This leads to a straightforward interactive rendering algorithm. Pseudo-code for the algorithm is given here. It leverages texture mapping graphics hardware in two ways: once to perform the warping and blending between the sample images; and again using the generated sphere map in the final rendering.

```
main()
// Set up radiance maps and sphere geometry
G_v ← LoadGeodesicLocations(); v ∈ {(θ_0, φ_0)...(θ_{N-1}, φ_{N-1})}
L_{r,j} ← LoadSphereMaps( G, L_i, f_r ); j ∈ {0...N − 1}
// Viewpoint tracking loop
loop for each frame
    (x_d, y_d, v_d) ← ComputeViewCoordinateSystem( )
    (v_0, v_1, v_2) ← FindSubtendedTriangle( G, v_d )
    (a_0, a_1, a_2) ← ComputeWeights( (v_0, v_1, v_2), v_d )
    glClearAccum( 0, 0, 0, 0 )
    // Warp and blending loop
    loop for each of the three vertices, i
        mesh ← ComputeMesh( v_i, v_d)
        drawMesh(mesh, L_{r,i})
        glAccum(GL_ACCUM, a_i)
    end vertex loop
    glAccum(GL_RETURN, 1.0/(a_0 + a_1 + a_2))
    LoadNewSphereMap()
    RenderObject()
end frame loop
```

The interactive rendering program outlined above reads in a set of sphere maps at a prescribed set of geodesic locations along with the associated triangle faces. This decouples the interactive program from any specific choice of sphere map sampling view directions.

5 EXAMPLES

We have validated our technique with several examples. One is shown in figure 3. This example shows the recently introduced Mercedes-Benz S-Class automobile in an outdoor environment. The car shows an isotropic BRDF modeling automobile paint, computed using techniques similar to that found in Gondek [9]. We modeled the clear polyurethane coat over a paint substrate containing paint particles. Using a custom offline ray tracer we directly solve the lighting integral for each point in twelve pre-computed radiance environment maps. Thus the under coat of the paint and the clear coat are both modeled with high fidelity. Each sphere map takes several minutes to create. Figure 5 shows all of the textures used to render the car example.

Our software is available online[1]. On a Silicon Graphics$^{(R)}$ Onyx2^{TM} InfiniteReality2^{TM}, the interactive viewing program runs at a sustained frame rate of 20Hz.

6 CONCLUSION

Interactive photo-realistic rendering is difficult to achieve because solving the lighting integral is computationally expensive. We precompute this integral into several view-dependent radiance environment maps, each of which would allow realistic rendering of arbitrary geometry from one fixed viewpoint. We dynamically create new maps for new viewpoints by combining the precomputed maps using an application of IBR techniques in reflection space. The resulting algorithm is readily implementable on most texture mapping

[1] http://www.sgi.com/software/rsibr/

capable graphics hardware. This technique allows rendering quality similar to that presented in Debevec [5], but at interactive rates and from arbitrary viewpoints.

Some areas of future work to improve this technique are apparent. We would like to perform the reflection-space IBR warp on a per pixel basis. We would also like to extend the range of BRDFs that can be accurately rendered. For example, we could handle arbitrary isotropic BRDFs with multiple high-frequency lobes in multiple passes, though admittedly with a loss in interactive performance. We would decompose the BRDF using a basis of symmetric lobes. Each basis function would be independently integrated with the environment and warped in a separate pass. We would also like to handle anisotropic BRDFs.

A broader area of future research is opened by the idea of reflection-space IBR. Traditional IBR could not have achieved these results; it is limited to rendering geometry contained in the source images. Traditional rendering, even using radiance environment maps, could also not have achieved these results; it could not provide the viewpoint independence without a fast way to create new intermediate maps. By applying IBR to an intermediate image representation in traditional rendering, it becomes possible to produce new algorithms that combine the strengths of both.

7 ACKNOWLEDGMENTS

The authors would like to thank all the individuals who helped us with the writing and development of this work. We are very grateful for the incredible S-Class data set from Andreas Fischer at Diamler-Chrysler – he and DiamlerChrysler have been invaluable colleagues and partners in our research into interactive surface rendering. During the writing and review process the anonymous reviewers provided valuable insight and suggestions. Also, Peter Shirley and Gavin Miller helped with guidance and clarifications to refine many of the key ideas in the paper. Mark Peercy helped out with technical inspiration, ideas, and proofreading throughout the paper writing endeavor. We would also like to thank Gosia Kulczycka at General Motors for the numerous paint samples and feedback on the viability and quality of our technique. Greg Larson-Ward was extremely helpful with advice, technical proofreading and energy balanced synthetic images from RADIANCE. Thanks to Dany Galgani who provided the excellent figure 1 illustration on very short notice. The cubemaps were stitched together using the Stitcher$^{(R)}$ software provided by REALVIZ. And thanks to the crew at Lawrence Livermore National Lab for helping in the printing of the color plates.

References

[1] BLINN, J. F., AND NEWELL, M. E. Texture and reflection in computer generated images. *Communications of the ACM 19* (1976), 542–546.

[2] BRONSHTEIN, I., AND SEMENDYAYEV, K. *Handbook of Mathematics.* Van Nostrand Reinhold Company, 1985.

[3] CABRAL, B., MAX, N., AND SPRINGMEYER, R. Bidirectional reflection functions from surface bump maps. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), M. C. Stone, Ed., vol. 21, pp. 273–281.

[4] COOK, R. L., CARPENTER, L., AND CATMULL, E. The Reyes image rendering architecture. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), M. C. Stone, Ed., pp. 95–102.

[5] DEBEVEC, P. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH 98 Conference Proceedings* (July 1998), M. Cohen, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 189–198. ISBN 0-89791-999-8.

[6] DEBEVEC, P. E., AND MALIK, J. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997), T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 369–378. ISBN 0-89791-896-7.

[7] FARIN, G. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1990.

[8] GERSHBEIN, R., SCHRÖDER, P., AND HANRAHAN, P. Textures and radiosity: Controlling emission and reflection with texture maps. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 51–58. ISBN 0-89791-667-0.

[9] GONDEK, J. S., MEYER, G. W., AND NEWMAN, J. G. Wavelength dependent reflectance functions. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 213–220. ISBN 0-89791-667-0.

[10] GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. The lumigraph. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 43–54. held in New Orleans, Louisiana, 04-09 August 1996.

[11] GREENE, N. Applications of world projections. In *Proceedings of Graphics Interface '86* (May 1986), M. Green, Ed., pp. 108–114.

[12] HE, X. D., TORRANCE, K. E., SILLION, F. X., AND GREENBERG, D. P. A comprehensive physical model for light reflection. In *Computer Graphics (SIGGRAPH '91 Proceedings)* (July 1991), T. W. Sederberg, Ed., vol. 25, pp. 175–186.

[13] HEIDRICH, W., AND SEIDEL, H.-P. Realistic, hardware-accelerated shading and lighting. In *Proceedings of SIGGRAPH '99 (Los Angeles, California, August 8–13, 1999)* (Aug. 1999), Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press.

[14] IMMEL, D. S., COHEN, M. F., AND GREENBERG, D. P. A radiosity method for non-diffuse environments. In *Computer Graphics (SIGGRAPH '86 Proceedings)* (Aug. 1986), D. C. Evans and R. J. Athay, Eds., vol. 20, pp. 133–142.

[15] JENSEN, H. W., AND CHRISTENSEN, P. H. Efficient simulation of light transport in scenes with participating media using photon maps. In *SIGGRAPH 98 Conference Proceedings* (July 1998), M. Cohen, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 311–320. ISBN 0-89791-999-8.

[16] KAJIYA, J. T. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)* (Aug. 1986), D. C. Evans and R. J. Athay, Eds., vol. 20, pp. 143–150.

[17] LEVOY, M., AND HANRAHAN, P. Light field rendering. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 31–42. held in New Orleans, Louisiana, 04-09 August 1996.

[18] MCMILLAN, L., AND BISHOP, G. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH 95 Conference Proceedings* (Aug. 1995), R. Cook, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 39–46. held in Los Angeles, California, 06-11 August 1995.

[19] MILLER, G. S., AND HOFFMAN, C. R. Illumination and reflection maps: Simulated objects in simulated and real environments. In *SIGGRAPH '84 Advanced Computer Graphics Animation seminar notes*. July 1984.

[20] OPPENHEIM, A. V., AND SCHAFER, R. W. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[21] POULIN, P., AND FOURNIER, A. A model for anisotropic reflection. In *Computer Graphics (SIGGRAPH '90 Proceedings)* (Aug. 1990), F. Baskett, Ed., vol. 24, pp. 273–282.

[22] SEITZ, S. M., AND DYER, C. R. View morphing: Synthesizing 3D metamorphoses using image transforms. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 21–30. held in New Orleans, Louisiana, 04-09 August 1996.

[23] VEACH, E., AND GUIBAS, L. J. Metropolis light transport. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997), T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 65–76. ISBN 0-89791-896-7.

[24] VOORHIES, D., AND FORAN, J. Reflection vector shading hardware. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 163–166. ISBN 0-89791-667-0.

[25] WARD, G. J. The RADIANCE lighting simulation and rendering system. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 459–472. ISBN 0-89791-667-0.

Figure 3: Mercedes-Benz S-Class automobile in an outdoor environment.

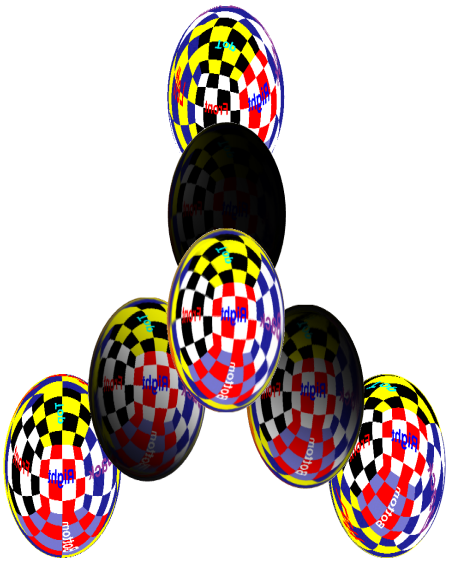

Figure 4: The outer images are source radiance environment maps for a test environment and a mirror BRDF. The next layer of images show each map warped to the new viewpoint with appropriate spherical barycentric weighting. The center image is the final generated radiance environment map.

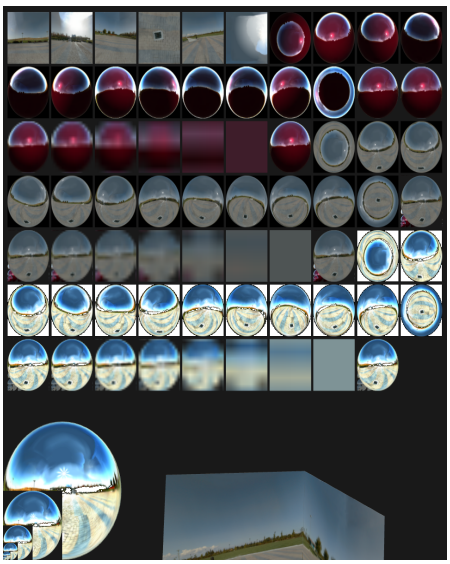

Figure 5: All of the textures used for figure 3. Includes the environment, source radiance environment maps for several surface types on the car, and generated MIP mapped radiance environment maps.