Reflection Space Image-Based Rendering (link to video)

Kevin Lim Myron Liu

Abstract

In this report we describe the implementation of Reflection Space Image-Base Rendering as described by [Cabral et al. 1999]. HDR environment maps contain high-quality lighting information, but integrating them is costly, especially if view-dependent materials are involved. Our goal is to generate a scene in which the user can rotate around the object lit by an HDR environment map, and observe view dependent BRDF changes in real-time.

Keywords: radiance, global illumination, real-time, environment maps, HDR

1 Motivation

Realistic scenes hardly ever consist purely of simple light sources. Environment maps capture complex lighting scenarios—such as environment lighting—from real scenes. If we assume that the environment is much farther from the objects than any length scale of the local scene, then we can store the lighting information in the form of an environment map; When stored this way, we can associate each point on the map with an incident light direction, and allow for complex lighting. However, there are challenges with doing this in real-time.

2 Challenges

For viewpoint independent BRDFs, it is a simple matter to achieve interactive framerates. By definition, viewpoint independence means that the BRDF $f(\theta_{in}, \phi_{in}; \theta_{out}, \phi_{out}) = f(\theta_{in}, \phi_{in})$ depends only on the incident directions. Under this condition, for each fixed orientation of a surface element (relative to the environment), the integration of irradiance over the upper hemisphere of the patch only needs to be done once. Following this computation, we can use the resulting radiance value for arbitrary viewpoint. This is what we do for Lambertian surfaces as shown in the results.

A similar simplification can be made for models such as Phong specular reflection. The argument is completely analogous. The only difference is that rather than index into the direction of the surface normal (which is what we did before), we index into the reflected direction (of the view vector about the surface normal). This sort of indexing into the reflected direction is what is meant by *reflection space* image based rendering. In either case, rendering is fast and easy, since the radiance depends only on two variables. In the former, the radiance is stored in terms of patch orientation (θ_n, ϕ_n). In the latter, the radiance is stored in terms of reflected view vector direction (θ_r, ϕ_r). We could, for instance, store this information in discretized form using a 2D θ, ϕ array.

Of course, perfectly mirrored reflection is even simpler. Since the BRDF in this case is a delta function in the reflected direction (given an incident direction), (θ_{in}, ϕ_{in}) and $(\theta_{out}, \phi_{out})$ are coupled, such that no integration even needs to be done, and a radiance map becomes redundant. Indexing into the environment map using the reflected direction suffices, since that is precisely the effect of integrating over the delta function.

Unfortunately, things aren't so simple when the BRDF is not viewpoint independent. In this case, we must store multiple radiance maps and interpolate between them. In the next section, we describe how to create the radiance maps, both for the simple case in which only one map is needed, and for the general, more-complicated case.

3 Creating the Radiance Map(s)

As previously mentioned, for viewpoint independent BRDFS, we only need one radiance map. For each point on the map (corresponding to each surface normal direction), we integrate over the upper hemisphere the environment intensity modulated by the BRDF and geometry terms (in this case, the cosine of the angle between incident light direction and patch normal), and index into the radiance map appropriately.

To handle 3D BRDFS that do not depend on ϕ_{in} we would need to stored a 4D array. This is because the space of viewing vectors is two dimensional, and for each, we need to store a two dimensional radiance map. The memory requirements to do this are astronomical, and we must resort to dramatically subsampling the space. Namely, we fix a handful of viewpoints and compute the radiance map for each such viewing direction. Then to obtain the appearance of a patch from an arbitrary viewing direction $\hat{v}_d$, we interpolate between the way the patch would appear from the nearest viewing directions $\hat{v}_0, \hat{v}_1, \hat{v}_2$. Of course, the number of viewpoints we need to precompute depends in part on how rapidly the BRDF varies. For something like Cook-Torrance, it suffices to precompute on viewpoints corresponding to the icosahedral directions. This provides adequate coverage of the sphere on which the view vector could lie. More precisely, if we let $R_i(\theta_n, \phi_n)$ be the radiance map corresponding to viewpoint $\hat{v}_i$ parameterized by surface normal orientation.

$$R_i(\hat{n}) = \int_{\theta_E} \int_{\phi_E} L_E(\theta_E, \phi_E) f(\theta_E, \phi_E; \hat{v}_i) \max(\hat{L}_E(\theta_E, \phi_E) \cdot \hat{n}, 0) \sin(\phi_E) d\theta_E d\phi_E$$

As alluded to above, this method cannot handle well BRDFs that depend on ϕ_{in}. Such materials appear differently when spun about its surface normal (brushed metal being one example). In this case, the full description is five dimensional, and introduces another layer of complexity. To apply the same methodology of saving a handful of 2D radiance maps (subsampling in the patch rotation in addition to subsampling in space of viewing directions) would be to take 2D slices of a 5D space, which is a far sparser sampling than before.

4 Interpolating Between Radiance Maps

Let $\hat{v}_0, \hat{v}_1, \hat{v}_2$ be the three nearest viewpoints. Supposing they are not colinear (which is guaranteed if we use the icosahedral directions), the reasonable way to combine the appearance is via spherical barycentric interpolation. The values we interpolate are simply the radiances for the material patch as it appears in the three maps *i.e.* we interpolate $R_0(\hat{n}), R_1(\hat{n}), R_2(\hat{n})$.

For each radiance map, we must define a local coordinate system $\{\hat{x}_i, \hat{y}_i, \hat{z}_i\}$ in which to store the precomputed values. A natural choice is to pick $\hat{z}_i = \hat{v}_i$; given $\hat{z}_i$, we pick $\hat{x}_i, \hat{y}_i$ accordingly while preserving handedness. In general, the choice of coordinate system is arbitary. All that matters is that a function g_i exists for each viewpoint that takes the representation of a point in the local coordinates of the i^{th} viewpoint to its representation in the global

coordinates of the environment map. If we pick right-handed, orthonormal $\{\hat{x}_i, \hat{y}_i, \hat{z}_i\}$, then g_i is just left matrix multiplication by $[\hat{x}_i|\hat{y}_i|\hat{z}_i]$.

So for *reflection space* (*e.g.* Cook Torrance Model), the radiance values $\boldsymbol{c}_i \equiv R_i(\hat{h}_i)$ we interpolate are obtained as follows (Also see Fig.1).

- Given $\hat{v}_d$ and $\hat{n}$, compute the reflected direction $\hat{r}$
- Find the three viewpoints nearest to $\hat{v}_d$: $\hat{v}_0, \hat{v}_1, \hat{v}_2$
- Find the half-angle $\hat{h}_i$ between $\hat{v}_i$. This is the patch orientation that would have given rise to $\hat{r}$ had we reflected $\hat{v}_i$ about $\hat{n}$
- Index into the local representation of $\hat{h}_i$ on the corresponding radiance map to obtain $\boldsymbol{c}_i \equiv R_i(\hat{h}_i)$.

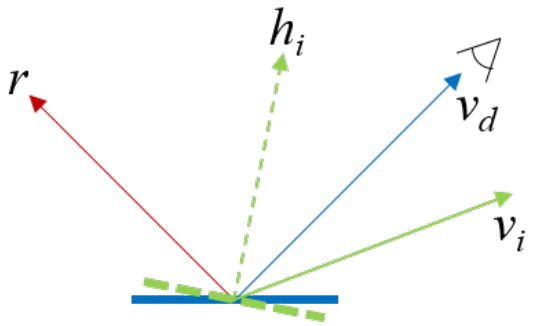


Figure 1: *The red arrow is the reflected direction given a surface element and viewing direction (blue). The green arrow is a nearby viewing direction for which we have precomputed the radiance map R_i. We index into the coordinate on R_i corresponding to the half angle $\hat{h}_i$, which is the patch orientation that would have given rise to $\hat{r}$ in the precomputed viewpoint.*

If we use a model that indexes into the direction of the surface normal, then the procedure is even simpler, since in this case, we need not compute $\hat{r}$ and $\hat{h}$. Instead, we index directly into the local representation of $\hat{n}$ in the precomputed map.

Each value to be interpolated is weighted using spherical barycentric coordinates (see Fig.2). The solid angle subtended by the triangle corresponding to $\hat{v}_i$ is

$$a_i = \alpha_i + \beta_i + \gamma_i - \pi$$

where α, β, γ are the dihedral angles

$$\alpha_i = \cos^{-1}\Big((\hat{v}_d \otimes \hat{v}_{i-1}) \cdot (\hat{v}_{i+1} \otimes \hat{v}_{i-1})\Big)$$
$$\beta_i = \cos^{-1}\Big((\hat{v}_{i+1} \otimes \hat{v}_d) \cdot (\hat{v}_{i+1} \otimes \hat{v}_{i-1})\Big)$$
$$\gamma_i = \cos^{-1}\Big((\hat{v}_{i+1} \otimes \hat{v}_d) \cdot (\hat{v}_{i-1} \otimes \hat{v}_d)\Big)$$

We have implicitly assumed index wrapping, such that $i-1$ and $i+1$ are actually $(i-1) \mod 3$ and $(i+1) \mod 3$. $\otimes$ is the normalized cross product *i.e.* $\boldsymbol{a} \otimes \boldsymbol{b} = (\boldsymbol{a} \times \boldsymbol{b})/\|\boldsymbol{a} \times \boldsymbol{b}\|$. Then the interpolated patch color is

$$\boldsymbol{c} = \frac{\sum_{i=0,1,2} a_i \boldsymbol{c}_i}{\sum_{i=0,1,2} a_i}$$

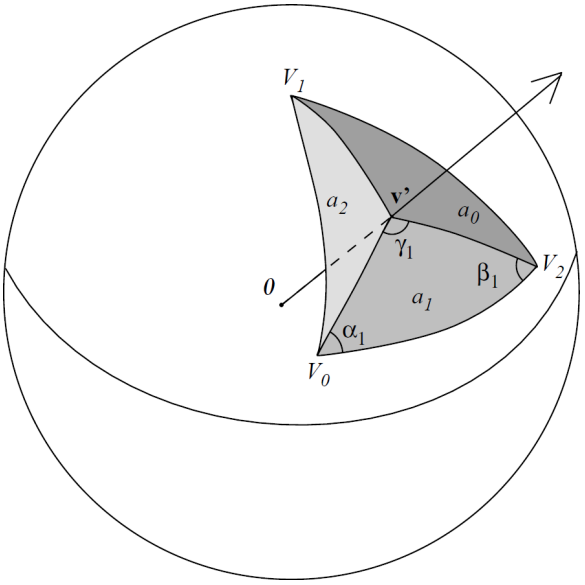


Figure 2: *$\hat{v}_d$ is the viewing direction, which can point in any direction. $\hat{v}_{0,1,2}$ are the nearest viewing directions on which the radiance map has already been precomputed. The appearance of a material element from the vantage point of $\hat{v}_d$ is obtained by averaging the way it would appear from viewing direction $\hat{v}_{0,1,2}$ each weighted by the solid angle subtended by the opposing triangle.*

5 Results

We've tested our implementation on an Intel i5 CPU, 8GB Memory, NVIDIA GeForce 560 Ti Graphics Card machine. Our Diffuse and Phong pre-integration took 2557.1 seconds and 3521.7 seconds for a 3072 pixels by 1536 pixels HDR Grace Cathedral map, provided by the USC HDR archive [Debevec and Malik 2008]. The Cook-Torrance integration was performed on a reduced map size of 100 pixels by 50 pixels, as our roughness value would be a high 0.1, and the shading will become blurred. This took an average 2.63 seconds per map, and for the twelve viewpoints, this took 31.6 seconds total.

Since shading information is pre-computed, the interactive scene can run at a smooth capped 60 frames per seconds for all four shader types. These figures are shown in figures 3, 4, 5, 6b, and 6a.

Acknowledgements

We would like to thank Professor Ravi Ramamoorthi for the amazing class and project.

References

CABRAL, B., OLANO, M., AND NEMEC, P. 1999. Reflection space image based rendering. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 165–170.

DEBEVEC, P. E., AND MALIK, J. 2008. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 2008 classes*, ACM, 31.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 497–500.

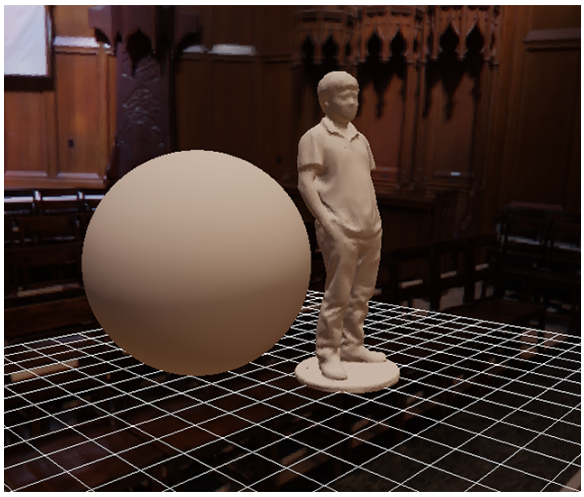
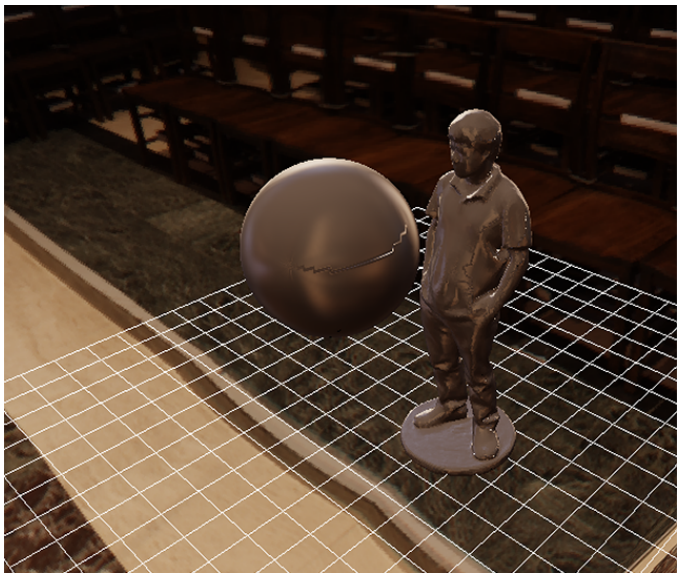


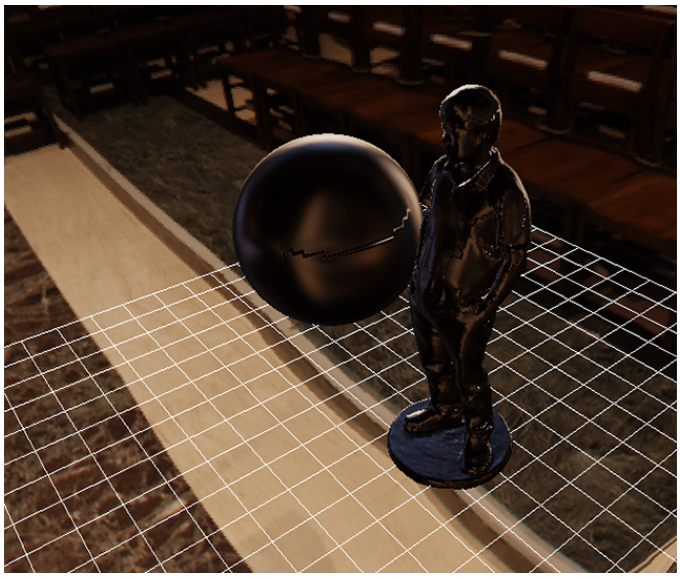
Figure 3: *Diffuse shader*

Figure 4: *Perfectly mirrored shader*

Figure 5: *Phong shader*



(a) *Cook-Torrance shader*



(b) *Cook-Torrance specular component shader*

Figure 6: *Cook-Torrance shaders*