Environment Mapping with Floyd-Steinberg Halftoning

László Szirmay-Kalos, László Szécsi, and Anton Penzov

In many computer graphics applications we wish to augment virtual objects with images representing a real environment (sky, city, etc.). In order to provide the illusion that the virtual objects are parts of the real scene, the illumination of the environment should be taken into account when rendering the virtual objects [Debevec 98, Kollig and Keller 03]. Since the images representing the environment lack depth information, we usually assume that the illumination stored in these images comes from far surfaces. This means that the illumination of the environment is similar to directional lights, it has only directional characteristics, but its intensity is independent of the location of the illuminated point.

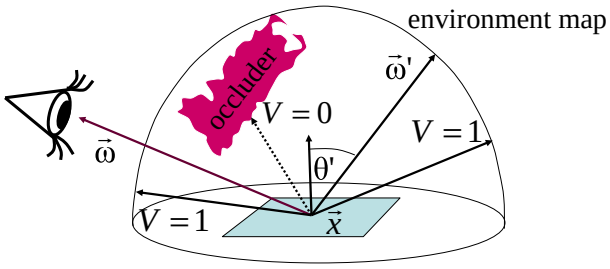


Figure 1.1. The concept of environment mapping stored in a cube map.

Environment mapping may be used to compute the reflected radiance of a *shaded point* $\vec{x}$ in *viewing direction* $\vec{\omega}$ as a directional integral

$$L(\vec{x}, \vec{\omega}) = \int_{\Omega} L^{\text{env}}(\vec{\omega}') f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cos\theta'_{\vec{x}} V(\vec{x}, \vec{\omega}') \mathrm{d}\omega',$$

where Ω is the set of all incident directions, $L^{\text{env}}(\vec{\omega}')$ is the radiance of the environment map at *illumination direction* $\vec{\omega}'$, f_r is the BRDF, $\theta'_{\vec{x}}$ is the angle between illumination direction $\vec{\omega}'$ and the surface normal at

$\vec{x}$, and $V(\vec{x}, \vec{\omega}')$ is the indicator function checking whether environment illumination can take effect in shaded point $\vec{x}$ at direction $\vec{\omega}'$, i.e. no virtual object occludes the environment in this direction (Figure 1.1).

This integral is usually estimated by Monte Carlo quadrature, which generates M number of samples $\vec{\omega}'_1, \ldots, \vec{\omega}'_M$ with probability density $p(\vec{\omega}')$ and computes the estimate as an average:

$$L(\vec{x}, \vec{\omega}) \approx \frac{1}{M} \sum_{i=1}^{M} \frac{L^{\text{env}}(\vec{\omega}'_i) f_r(\vec{\omega}'_i, \vec{x}, \vec{\omega}) \cos\theta'_{\vec{x},i} V(\vec{x}, \vec{\omega}'_i)}{p(\vec{\omega}_i)}.$$

The most time consuming part of the evaluation of a sample is the computation of visibility factor V, i.e. the determination whether or not the environment is occluded. Real-time environment mapping algorithms usually ignore this factor and consequently the shadowing of environment lighting. However, for rendering photo-realistic images, this simplification is unacceptable. Thus, in this article we examine environment mapping approaches that correctly evaluate the occlusion of the illuminating environment.

The calculation of the visibility factor requires tracing a ray originating at shaded point $\vec{x}$ and having direction $\vec{\omega}'_i$. In order to improve the speed of environment mapping with shadow computation, the number of samples, that is the number of traced rays, should be minimized.

For a given number of samples, the error of the quadrature depends on two factors:

1. *Importance sampling*: How well does density p mimic the integrand?

2. *Stratification*: How well does the empirical distribution of the finite number of samples follow the theoretical distribution defined by p?

This means that we should do a good job in both mimicking the integrand with the sample density and producing well stratified samples.

1.1 Parametrization of the environment map

The environment illumination is defined by a texture map $T(u, v)$ addressed by texture coordinates $u, v \in [0, 1]$. Thus, we need a mapping or a parametrization that defines the correspondence between a texture coordinate pair and illuminating direction $\vec{\omega}'$.

A possible parametrization expresses direction $\vec{\omega}'$ by spherical angles θ', ϕ' where $\phi' \in [0, 2\pi]$ and $\theta' \in [0, \pi/2]$ in the case hemispherical lighting and $\theta' \in [0, \pi]$ in the case of spherical lighting. Then texture coordinates

(u, v) are scaled from the unit interval to these ranges. For example, in the case of spherical lighting, a direction is parameterized as

$$\vec{\omega}'(u, v) = (\cos 2\pi u \sin \pi v,\ \sin 2\pi u \sin \pi v,\ \cos \pi v),$$

where $u, v \in [0, 1]$.

A texture map is a two-dimensional image containing $R_u \times R_v$ texels where R_u and R_v are the horizontal and vertical resolutions, respectively. Note that the discussed parametrization is not uniform since different texels correspond to the same $\Delta u \Delta v = (1/R_u)(1/R_v)$ area in texture space, but different solid angles $\Delta\omega$ depending on texture coordinate v:

$$\Delta\omega = \sin \pi v \Delta u \Delta v = \frac{\sin \pi v}{R_u R_v}.$$

The integral of the reflected radiance can also be evaluated in texture space:

$$L(\vec{x}, \vec{\omega}) = \int_{u=0}^{1} \int_{v=0}^{1} E(u, v) R(u, v, \vec{x}, \omega) \mathcal{V}(u, v, \vec{x}) \mathrm{d}v \mathrm{d}v \approx$$

$$\frac{1}{R_u R_v} \sum_{i=1}^{R_u} \sum_{j=1}^{R_v} E\left(\frac{i}{R_u}, \frac{j}{R_v}\right) R\left(\frac{i}{R_u}, \frac{j}{R_v}, \vec{x}, \vec{\omega}\right) \mathcal{V}\left(\frac{i}{R_u}, \frac{j}{R_v}, \vec{x}\right),$$

where we used the following shorthand notations for the three main factors of the integrand:

$$E(u, v) = L^{\mathrm{env}}(\vec{\omega}'(u, v)) \sin \pi v = T(u, v) \sin \pi v$$

is the intensity of the *environment lighting* taking into account the distortion of the parametrization,

$$R(u, v, \vec{x}, \omega) = f_r(\vec{\omega}'(u, v), \vec{x}, \vec{\omega}) \cos \theta'_{\vec{x}}(u, v)$$

is the *reflection factor*, and

$$\mathcal{V}(u, v, \vec{x}) = V(\vec{x}, \vec{\omega}'(u, v))$$

is the *visibility factor*.

The evaluation of the reflected radiance by adding the contribution of all texels would be too time consuming. Therefore, we apply Monte Carlo methods, which approximate it from just a few sample directions, i.e. a few texels.

1.2 Importance sampling

Monte Carlo methods use a probability density to select the sample points. According to the concept of importance sampling, we should find a density p that mimics the product form integrand. To define an appropriate density, we usually execute the following three main steps.

1. First we decide which factors of the product form integrand will be mimicked and find a scalar approximation of the usually vector valued integrand factor. In our case, environment lighting E and reflection factor R are vector valued since they assign different values for the wavelengths of the red, green, and blue light. Spectrum L can be converted to a scalar by obtaining the *luminance* of the spectrum $\mathcal{L}(L)$, which is a weighted sum of the red, green, and blue intensities. The resulting scalar approximation of the integrand is called the *importance function* and is denoted by I. Note that as the environment illumination is defined by a texture, the importance function is also represented by a two-dimensional image. When we want to emphasize this property, we refer to the importance function as the *importance map*.

   There are several options to define the importance function as there are different alternatives of selecting those factors of the integrand that are mimicked. The simplest way is *BRDF sampling* that mimics the luminance of the reflection factor. *Light source sampling*, on the other hand, sets the importance function to be the luminance of the environment lighting. Finally, *product sampling* includes more than one factor of the integrand into the importance function. For example, the importance function can be the luminance of the product of the environment lighting and the reflection factor (double product sampling), or it can even incorporate a cheap visibility factor approximated by some simple proxy geometry included in the object (triple product sampling).

2. As the density should be normalized, the integral of the importance function needs to be computed for the whole domain. This computation can take advantage of the fact that the importance function is defined also as a two-dimensional array or a texture similarly to the texture map of the environment illumination:

$$\int_{u=0}^{1}\int_{v=0}^{1} I(u,v)\mathrm{d}v\mathrm{d}u \approx \frac{S}{R_u R_v}$$

where

$$S = \sum_{u=1}^{R_u} \sum_{j=1}^{R_v} I\left(\frac{i}{R_u}, \frac{j}{R_v}\right)$$

is the sum of all values in the importance map. Note that the importance function should be integrated as one step of the importance sampling. It means that the importance function must be much cheaper to evaluate and integrate than the original integrand.

3. Finally, the density is defined as the ratio of the importance function and the normalization constant:

$$p(u,v) = \frac{I(u,v)}{\int_{u=0}^{1} \int_{v=0}^{1} I(u,v)\mathrm{d}v\mathrm{d}u} = \frac{I(u,v)R_u R_v}{S}.$$

Having constructed the importance map and computed sum S, samples can be drawn with probability density p using the following simple method. We generate M statistically independent random numbers $r_1, \ldots, r_M$ that are uniformly distributed in the unit interval. Then, for each random number, the two-dimensional array of the importance map is scanned, and the importance values are added together. This running sum is compared to $r_i S$, and when the running sum gets larger, the scanning is stopped and the current texel (i.e. the direction corresponding to this texel) is considered as a sample. As can be shown easily, the process will select a texel with a probability that is proportional to its value.

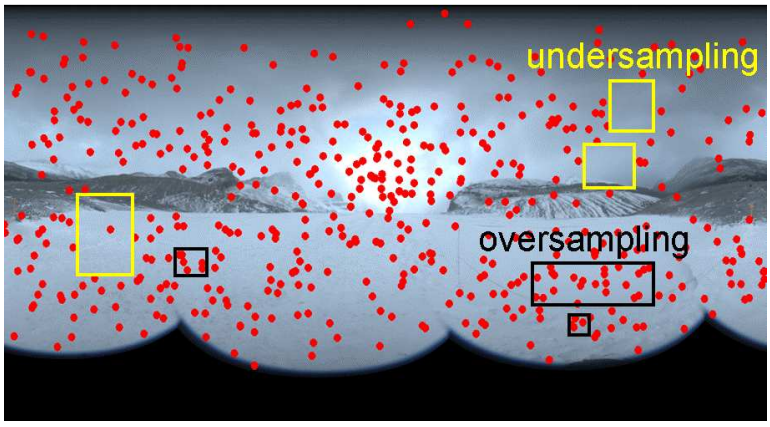


Figure 1.2. Environment map with random light source sampling.

Unfortunately, the application of statistically independent random samples provides poor results in many cases. To demonstrate the problem, we used light source sampling to find directional samples on an environment map (Figure 1.2). The results are disappointing since making independent random texel selections, with probability in proportion to its luminance, does not guarantee that groups of samples will be well stratified. We still have large empty regions in important parts of the map (*undersampling*) and groups of samples needlessly concentrating to a small unimportant region (*oversampling*). We note that this problem has also been addressed by Kollig who proposed the relaxation of the samples [Kollig and Keller 03] and by Ostromoukhov who applied sophisticated tiling [Ostromoukhov et al. 04]. The method proposed in the next section provides similar results as these methods, but is much simpler and has practically no overhead with respect to the simple random approach.

1.3 Proposed solution

The method proposed in this article has the goal of producing well stratified samples mimicking an *importance map*. It is effective, simple to implement and is even faster than random sampling.

gray-scale image | random halftoning | Floyd-Steinberg halftoning

Figure 1.3. A gray-scale image and its halftoned versions obtained with random halftoning and with the Floyd-Steinberg algorithm.

The proposed method is based on the recognition that importance sampling is equivalent to *digital halftoning* [Szirmay-Kalos et al. 09]. Halftoning is a technique used to render gray-scale images on a black and white display. The idea is to put more white points at brighter areas and fewer points at darker areas. The spatial density of white points in a region around a pixel is expected to be proportional to the gray level of that particular

pixel. If we consider the gray level of the original image to be an importance function and the white pixels of the resulting image to be sample locations, then we can see that halftoning is equivalent to a deterministic importance sampling algorithm. The equivalence of importance sampling and halftoning stems from the fact that both of them are *frequency modulators* [Szirmay-Kalos and Szécsi 09]. The input of the frequency modulator is the gray-scale image or the importance map, respectively, and the output is a collection of discrete samples with a frequency specified by the input.

This equivalence holds for an arbitrary halftoning algorithm, including the random and ordered halftoning methods that add random noise or a periodic pattern to the original image before quantization, or, for example, error diffusion halftoning methods from which the Floyd-Steinberg algorithm is the most famous [Floyd and Steinberg 75]. Error diffusion halftoning provides better results than random or ordered halftoning, because it does not simply make independent local decisions, but gathers and distributes information to neighboring pixels as well. Because it takes gray levels in a neighborhood into account, the sample positions are stratified, making the resulting image smoother and reducing the noise compared to random or dithered approaches.

Due to these nice properties, we developed our sampler based on the Floyd-Steinberg method. Random dithering was implemented for comparison. We expected the same improvement in importance sampling as provided by the Floyd-Steinberg halftoning over random dithering.

1.3.1 Floyd-Steinberg sampler

The sampling algorithm takes the importance map and computes the sum S of all texels. Then, the sampling is simply the execution of a Floyd-Steinberg halftoning on the map setting the threshold at $S/(2M)$ where M is the number of expected samples. In Figure 1.4 the threshold and the error are depicted by a red line and a white bar, respectively. The halftoning algorithm initializes an error value to 0 and scans the map row-by-row changing the scanning order at the end of the rows. At each texel, the comparison of the error value to the threshold may have two outcomes. If the error is not greater than the threshold, then no sample is generated here (the texel becomes black) and the error is left unchanged. If the error is greater than the threshold, then this texel is a sample, and the error value is decreased by S/M, i.e. we compute the negative complementer of the error represented by the black part of the bar in Figure 1.4.

In both cases, before stepping onto the next texel, the remaining error of the texel is distributed to its unvisited neighbors. The method continues similarly until all texels have been visited.

Listing 1.1 shows the implementation of this algorithm optimized to

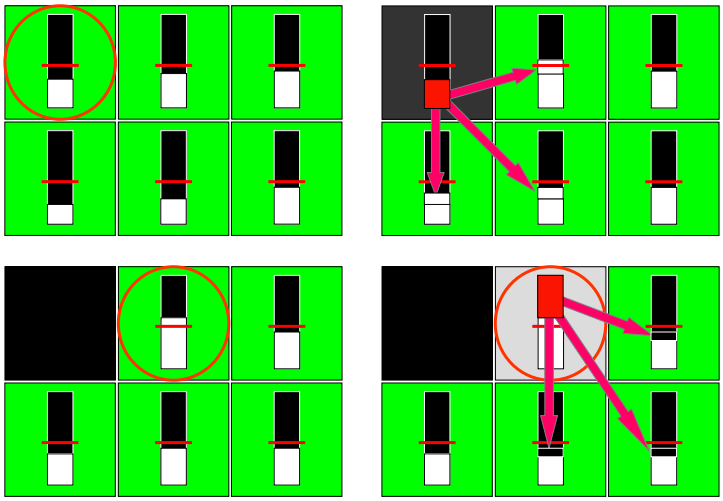


Figure 1.4. The Floyd-Steinberg sampling. The error (white bar) of the upper-left texel is smaller than the threshold (red line), so it is not selected (the texel will be black), and its error is distributed to the neighbors increasing their error levels. The next texel will have error larger than the threshold, so it is selected (the texel will be white), and the negative complementer error (black bar) is added to the neighbors reducing their error value.

work as a geometry shader. Every time the shader is invoked, it processes the importance map of size `R.x` × `R.y`, the values of which are queried using the `getImportance` function. It emits 32 directional samples, with the probability of sample selection stored in the alpha channel. The function `getSampleDir` returns the direction associated with a texel of the importance map. We avoid maintaining an actual array of importance values by storing only the importance that has been carried to the next row. Variable `cPixel` contains the importance to be transferred to the next pixel, `cDiagonal` must be added to the pixel below the next, and `cRow` is an array, packed into `float4` vectors, that contains the importances to be added to pixels in the next row. Every row is processed in runs of four pixels, after which the four values gathered in variable `acc` can be packed into the `cRow` array. The size of `cRow` is `RX4`, the width of the importance map divided by four. Every time the importance map is read, the original importance value is loaded into variable `I`, and importance carried over from the neighbors is added to get the modified importance in variable `Ip`.

```
[maxvertexcount(32)]
void gsSampler( inout PointStream<float4> samples ) {
  uint M = 32; float S = 0;
  [loop]for(uint v = 0; v < R.y; v++)
    [loop]for(uint u = 0; u < R.x; u++)
      S += getImportance(uint2(u, v));
  float threshold = S / 2 / M;
  float4 cRow[RX4]={{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
  float cPixel = 0, cDiagonal = 0, acc[4];
  [loop]for(uint j = 0; j < R.y; j++)  {
    uint kper4 = 0;
    [loop]for(uint k = 0; k < R.x; k += 4) {
      for(uint xi = 0; xi < 4; xi++) {
        float I = getImportance(uint2(k+xi, j));
        float Ip = I + cRow[kper4][xi] + cPixel;
        if(Ip > threshold) {
          float3 dir = getSampleDir(uint2(k+xi, j));
          samples.Append( float4(dir, I / S) );
          Ip -= threshold * 2;
        }
        acc[xi] = Ip * 0.375 + cDiagonal;
        cPixel = Ip * 0.375;
        cDiagonal = Ip * 0.25;
      }
      cRow[kper4++] = float4(acc[0], acc[1], acc[2], acc[3]);
    }
    j++; kper4--;
    [loop]for(int k = R.x-5; k >= 0; k -= 4) {
      for(int xi = 3; xi >= 0; xi--) {
        float I = getImportance(uint2(k+xi, j));
        float Ip = I + cRow[kper4][xi] + cPixel;
        if(Ip > threshold ) {
          float3 dir = getSampleDir(uint2(k+xi, j));
          samples.Append( float4(dir, I / S) );
          Ip -= threshold * 2;
        }
        acc[xi] = Ip * 0.375 + cDiagonal;
        cPixel = Ip * 0.375;
        cDiagonal = Ip * 0.25;
      }
      cRow[kper4--] = float4(acc[0], acc[1], acc[2], acc[3]);
    }
  }
}
```

Listing 1.1. The Floyd-Steinberg sampler implemented as a geometry shader.

In addition to Floyd-Steinberg halftoning, the family of error diffusion methods has many other members that differ in the error distribution neighborhood and weights, and also in the order of processing the texels [Kang 99]. These sophisticated techniques are also worth using as importance sampling methods. The weights need special care when the number of expected samples M is very small with respect to the number of texels. Multi-dimensional error diffusion methods perform well when they are in a stationary state, but need to warm up, i.e. they start producing samples later than expected. If very few samples are generated with the method, this delay becomes noticeable. To solve this problem, the weight of the faster running coordinate should be increased, and in the extreme case, the algorithm should act as a one-dimensional error diffusion filter.

1.3.2 Application to light source sampling

In light source sampling the importance function is based on the environment illumination and we also take into account that different texels correspond to different solid angles:

$$I(u, v) = \mathcal{L}(E(u, v)).$$

Ignoring the cosine weighted BRDF and the visibility degrades importance sampling, but, as the importance function depends just on the illumination direction and is independent of the point being shaded, $\vec{x}$, sampling should be executed only once for all shaded points.

In this case, the proposed scheme is as simple as the Floyd-Steinberg halftoning of the environment map weighted by solid angle scaling $\sin \pi v$. Figures 1.5 and 1.6 compare the distribution of samples and the resulting images of the illuminated object for random sampling and for the Floyd-Steinberg sampler. As the Floyd-Steinberg sampler scans the map only once and obtains samples directly, it is not slower than random sampling. Sampling the 1024×512 resolution environment map of Figure 1.5 takes 47 msec on an nVidia GeForce 8800 GFX GPU.

1.3.3 Application to product sampling

Product sampling includes more than one factor of the integrand into the importance function. Note that the inclusion of all factors is not feasible since the computation of the importance must be cheaper than that of the integrand. In environment mapping, the expensive part is the visibility test, so we either ignore occlusions in the importance or replace it with some cheaper approximation. The importance function is defined as

$$I(u, v) = E(u, v)R(u, v, \vec{x}, \vec{\omega})\tilde{\mathcal{V}}(u, v, \vec{x}).$$

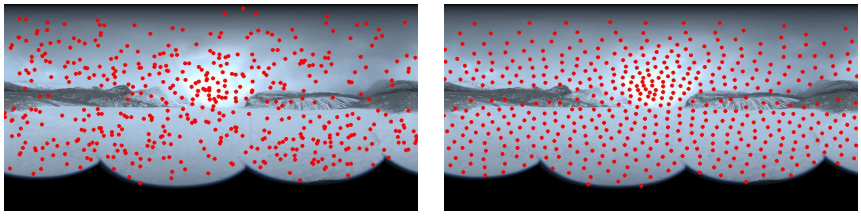
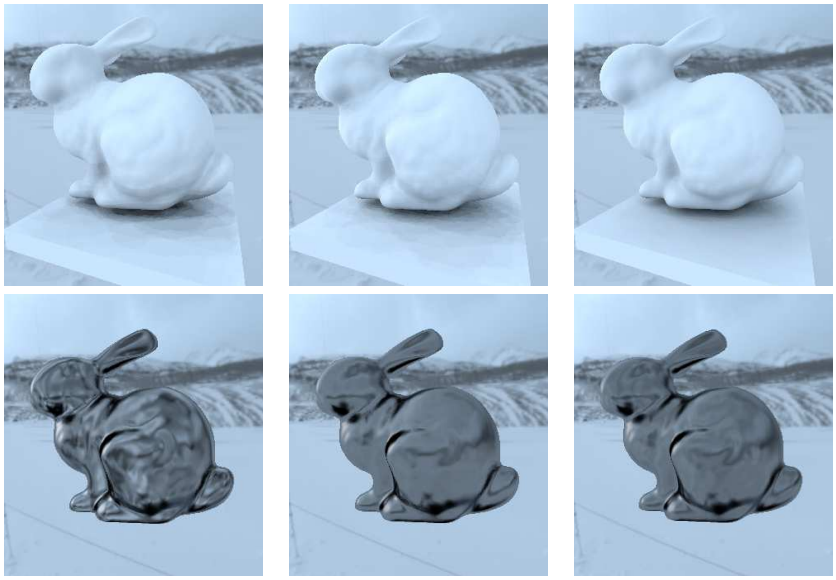


Figure 1.5. Sampling weighted environment maps with random sampling (left) and Floyd-Steinberg halftoning (right).



Random sampling Floyd-Steinberg Reference

Figure 1.6. Results of light source sampling. Diffuse and specular bunnies illuminated by directional lights sampled randomly and with Floyd-Steinberg halftoning.

where $\tilde{\mathcal{V}}$ is the approximation of the visibility factor. *Double product sampling* sets $\tilde{\mathcal{V}} = 1$ assuming that the environment is always visible when generating important directions. Alternatively, we can approximate visibility by computing intersections with a contained proxy geometry, for example, spheres inside the object. In this case, we talk about *triple prod-*

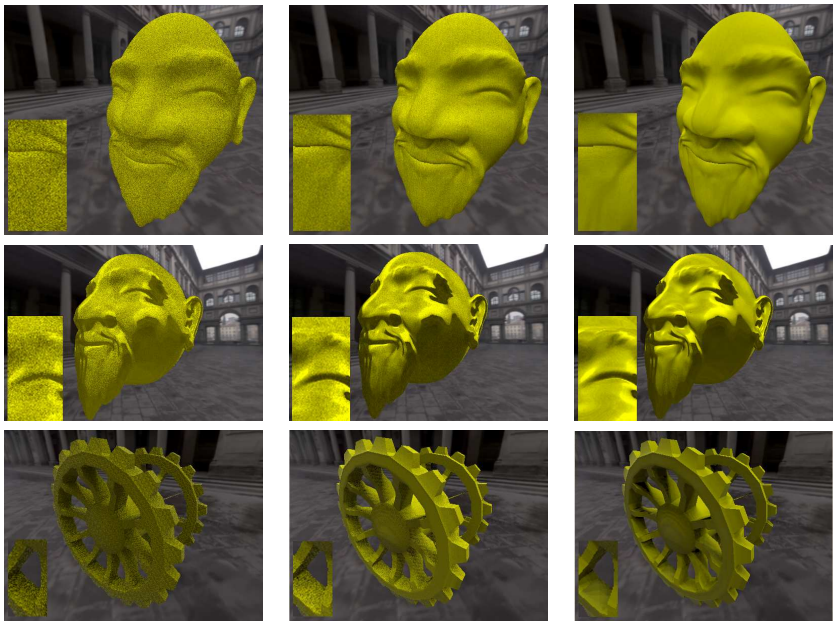
uct sampling. We have to emphasize that the approximate visibility factor and the proxy geometry is used only in the definition of the importance map and for generating important directions. When the ray is traced, the original geometry is intersected, that is, the original visibility indicator is included into the integral quadrature. Triple product sampling helps to reduce the number of those rays that surely intersect some object, and thus their contribution is zero. Unfortunately, it is not always easy to find a simple proxy geometry that is inside the object. For example, in Figure 1.7 it is straightforward to put a sphere into the Ming head, but the definition of a proxy geometry for the wheel is difficult.

Unlike light source sampling, now the importance function also depends on shaded point $\vec{x}$ and indirectly on the normal vector at $\vec{x}$. This means that we cannot process the environment map once globally for all shaded points, but the sampling process including the Floyd-Steinberg halftoning should be repeated for every single shaded point. Thus, while in light source sampling the Floyd-Steinberg sampler has no overhead, product sampling pays off if ray tracing is more costly than the generation and processing of the importance map.

In order to test the approach, we have compared three techniques: BRDF sampling, random halftoning, which is similar to *sampling-importance resampling* (SIR) [Burke et al. 04, Talbot et al. 05] in the case of product sampling, and the new Floyd-Steinberg scheme. All three were implemented as GPU algorithms, which run on nVidia GeForce 8800 GFX graphics hardware. All methods traced $M = 32$ rays per pixel. Both sampling-importance resampling and the Floyd-Steinberg sampler obtained the real samples from 32×32 local importance maps generated separately for every shaded point $\vec{x}$. The results are shown by Figure 1.7. Note that the Floyd-Steinberg sampler completely eliminated the noise at fully visible surfaces and some noise remained only at partially occluded regions.

1.4 Conclusions

The most important message of this article is that halftoning and importance sampling are equivalent, thus we can exploit the sophisticated halftoning algorithms in importance sampling. We investigated the application of the Floyd-Steinberg halftoning method in environment mapping and concluded that this approach produces samples with better distribution than random sampling. Thanks to this, the integrals evaluated with these samples are more accurate.



BRDF sampling SIR Floyd-Steinberg

Figure 1.7. Double product sampling results. Note that the Floyd-Steinberg sampler eliminated the noise at fully visible surfaces both for the diffuse and specular cases. The lower row of images show a wheel having a lot of occlusions, which are not mimicked by the double product importance.

Bibliography

[Burke et al. 04] David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. "Bidirectional importance sampling for illumination from environment maps." In *ACM SIGGRAPH 2004 Sketches*, p. 112, 2004.

[Debevec 98] Paul Debevec. "Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography." In *SIGGRAPH '98*, pp. 189–198, 1998.

[Floyd and Steinberg 75] Robert W. Floyd and Louis Steinberg. "An Adaptive Algorithm for Spatial Gray Scale." In *Society for Information Display 1975 Symposium Digest of Tecnical Papers*, p. 36, 1975.

[Kang 99] Henry R. Kang. *Digital Color Halftoning*. SPIE Press, 1999.

[Kollig and Keller 03] Thomas Kollig and Alexander. Keller. "Efficient Illumination by High Dynamic Range Images." In *Eurographics Symposium on Rendering*, pp. 45–51, 2003.

[Ostromoukhov et al. 04] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. "Fast Hierarchical Importance Sampling with Blue Noise Properties." *ACM Transactions on Graphics* 23:3 (2004), 488–498.

[Szirmay-Kalos and Szécsi 09] László Szirmay-Kalos and László Szécsi. "Deterministic Importance Sampling with Error Diffusion." *Computer Graphics Forum (EG Symposium on Rendering)* 28:4 (2009), 1056–1064.

[Szirmay-Kalos et al. 09] László Szirmay-Kalos, László Szécsi, and Anton Penzov. "Importance Sampling with Floyd-Steinberg Halftoning." In *Eurographics 09, Short papers*, pp. 69–72, 2009.

[Talbot et al. 05] Justin Talbot, David Cline, and Parris K. Egbert. "Importance Resampling for Global Illumination." In *Rendering Techniques*, pp. 139–146, 2005.