Monte Carlo Integration II

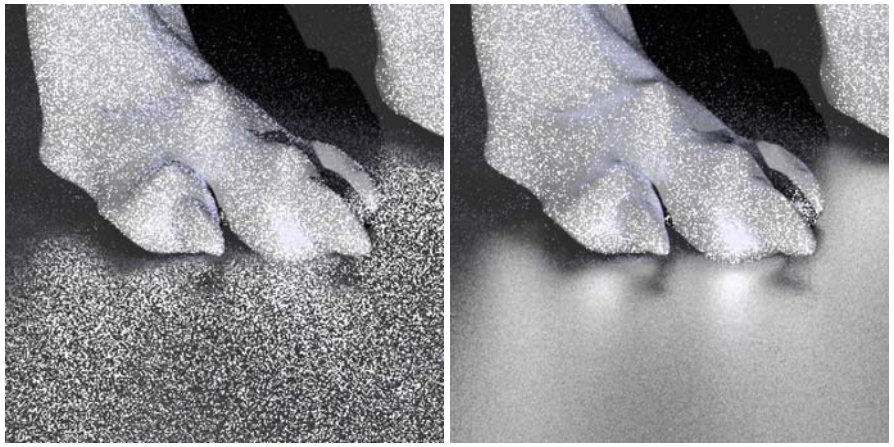
Digital Image Synthesis

Yung-Yu Chuang

12/7/2006

with slides by Pat Hanrahan and Torsten Moller

Variance = noise in the image



without variance reduction | with variance reduction

Same amount of computation for rendering this scene with glossy reflection

Variance reduction

- Efficiency measure for an estimator

$$Efficiency \propto \frac{1}{Variance \bullet Cost}$$

- If one technique has twice the variance as another technique, then it takes twice as many samples to achieve the same variance
- If one technique has twice the cost of another technique with the same variance, then it takes twice as much time to achieve the same variance
- Although we call them variance reduction techniques, they are actually techniques to **increase efficiency**
  - Importance sampling
  - Stratified sampling

Russian roulette and splitting

- Select some termination probability q,

$$F' = \begin{cases} \frac{F-qc}{1-q} & \xi > q \\ c & \text{otherwise} \end{cases}$$

$$E[F'] = (1-q)\left(\frac{E[F]-qc}{1-q}\right) + qc = E[F]$$

- Russian roulette will actually increase variance, but improve efficiency if q is chosen so that samples that are likely to make a small contribution are skipped.

Russian roulette and splitting

- Assume that we want to estimate the following direct lighting integral

$$L_o(p,\omega_o) = \int_\Omega f_r(p,\omega_o,\omega_i) L_d(p,\omega_i) |\cos\theta_i| \, d\omega_i$$

- The Monte Carlo estimator is

$$\frac{1}{N}\sum_{i=1}^{N} \frac{f_r(p,\omega_o,\omega_i) L_d(p,\omega_i) |\cos\theta_i|}{p(\omega_i)}$$

- Since tracing the shadow ray is very costly, if we somewhat know that the contribution is small anyway, we would like to skip tracing.
- For example, we could skip tracing rays if $|\cos\theta_i|$ or $f_r(p,\omega_o,\omega_i)$ is small enough.

Russian roulette and splitting

- Russian roulette makes it possible to skip tracing rays when the integrand's value is low while still computing the correct value on average.

Careful sample placement

- Carefully place samples to less likely to miss important features of the integrand
- Stratified sampling: the domain $[0,1]^s$ is split into strata $S_1..S_k$ which are disjoint and completely cover the domain.

$$S_i \cap S_j = \phi \quad i \neq j \qquad \bigcup_{i=1}^{k} S_i = [0,1]^s$$

$$|S_i| = v_i \qquad \sum v_i = 1$$

$$p_i(x) = \begin{cases} 1/v_i & \text{if } x \in S_i \\ 0 & \text{otherwise} \end{cases}$$

Stratified sampling

- Estimator for ith strata $\hat{I}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} f(X_{i,j})$
- Final estimator $\hat{I} = \sum_{i=1}^{k} v_i \hat{I}_i$

$$\mu_i = E[f(X_{i,j})] = \frac{1}{v_i}\int_{S_i} f(x)dx \qquad E[\hat{I}_i] = \mu_i$$

$$\sigma_i^2 = V[f(X_{i,j})] = \frac{1}{v_i}\int_{S_i} (f(x)-\mu_i)^2 dx \qquad V[\hat{I}_i] = \frac{\sigma_i^2}{n_i}$$

$$V[\hat{I}_s] = V\left[\sum_{i=1}^{k} v_i \hat{I}_i\right] = \sum_{i=1}^{k} V\left[v_i \hat{I}_i\right] = \sum_{i=1}^{k} v_i^2 V\left[\hat{I}_i\right] = \sum_{i=1}^{k} \frac{v_i^2 \sigma_i^2}{n_i} = \frac{1}{N}\sum_{i=1}^{k} v_i \sigma_i^2$$

assume that $n_i = v_i N$

Stratified sampling

$$V[f(X)] = \sum_{i=1}^{k} v_i \sigma_i^2 + \sum_{i=1}^{k} v_i (\mu_i - I)^2$$

because $V[X] = E[V[X \mid Y]] + V[E[X \mid Y]]$

$$\begin{aligned} V[X] &= E[X^2] - E[X]^2 \\ &= E[E[X^2 \mid Y]] - E[E[X \mid Y]]^2 \\ &= E[E[X^2 \mid Y]] - E[E[X \mid Y]^2] + E[E[X \mid Y]^2] - E[E[X \mid Y]]^2 \\ &= E[E[X^2 \mid Y] - E[X \mid Y]^2] + (E[E[X \mid Y]^2] - E[E[X \mid Y]]^2) \\ &= E[V[X \mid Y]] + V[E[X \mid Y]] \end{aligned}$$

Y represents the event of choosing strata and X represents the event of choosing within strata

$$V[\hat{I}_{ns}] = \frac{1}{N}\left[\sum_{i=1}^{k} v_i \sigma_i^2 + \sum_{i=1}^{k} v_i (\mu_i - I)^2\right]$$

Stratified sampling

$$V[\hat{I}_s] = \frac{1}{N}\sum_{i=1}^{k} v_i \sigma_i^2$$

$$V[\hat{I}_{ns}] = \frac{1}{N}\left[\sum_{i=1}^{k} v_i \sigma_i^2 + \sum_{i=1}^{k} v_i (\mu_i - I)^2\right]$$

Thus, the variance can only be reduced by using stratified sampling.

Stratified sampling

without stratified sampling

with stratified sampling

Bias

- Another approach to reduce variance is to introduce bias into the computation.

$$\beta = E[F] - \int f(x)dx$$

- Example: estimate the mean of a set of random numbers X_i over [0..1].

unbiased estimator $\frac{1}{N}\sum_{i=1}^{N} X_i$ variance (N^{-1})

biased estimator $\frac{1}{2}\max(X_1, X_2, ..., X_N)$ variance (N^{-2})

Pixel reconstruction

$$I = \int w(x) f(x) dx$$

- Biased estimator $\hat{I}_b = \frac{\sum_{i=1}^{N} w(X_i) f(X_i)}{\sum_{i=1}^{N} w(X_i)}$ (but less variance)
- Unbiased estimator $\hat{I}_u = \frac{\sum_{i=1}^{N} w(X_i) f(X_i)}{N p_c}$

where p_c is the uniform PDF of choosing Xi

$$E[\hat{I}_u] = \frac{1}{N p_c} \sum_{i=1}^{N} E[w(X_i) f(X_i)]$$

$$= \frac{1}{N p_c} \sum_{i=1}^{N} \int w(x) f(x) p_c dx = \int w(x) f(x) dx$$

Importance sampling

- The Monte Carlo estimator

$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}$$

converges more quickly if the distribution $p(x)$ is similar to $f(x)$. The basic idea is to concentrate on where the integrand value is high to compute an accurate estimate more efficiently.

- So long as the random variables are sampled from a distribution that is similar in shape to the integrand, variance is reduced.

Informal argument

- Since we can choose $p(x)$ arbitrarily, let's choose it so that $p(x) \sim f(x)$. That is, $p(x)=cf(x)$. To make $p(x)$ a pdf, we have to choose c so that

$$c = \frac{1}{\int f(x) dx}$$

- Thus, for each sample X_i, we have

$$\frac{f(X_i)}{p(X_i)} = \frac{1}{c} = \int f(x) dx$$

Since c is a constant, the variance is zero!

- This is an ideal case. If we can evaluate c, we won't use Monte Carlo. However, if we know $p(x)$ has a similar shape to $f(x)$, variance decreases.

Importance sampling

- Bad distribution could hurt variance.

$$I = \int_0^4 x dx = 8$$

method	Sampling function	variance	Samples needed for standard error of 0.008
importance	(6-x)/16	56.8/N	887,500
importance	1/4	21.3/N	332,812
importance	(x+2)/16	6.4/N	98,432
importance	x/8	0	1
stratified	1/4	$21.3/N^3$	70

Importance sampling

- Fortunately, it is not too hard to find good sampling distributions for importance sampling for many integration problems in graphics.
- For example, in many cases, the integrand is the product of more than one function. It is often difficult construct a pdf similar to the product, but sampling along one multiplicand is often helpful.

Multiple importance sampling

- To estimate $\int f(x)g(x)dx$, MIS draws n_f samples according to p_f and n_g samples to p_g, The Monte Carlo estimator given by MIS is

$$\frac{1}{n_f}\sum_{i=1}^{n_f}\frac{f(X_i)g(X_i)w_f(X_i)}{p_f(X_i)}+\frac{1}{n_g}\sum_{j=1}^{n_g}\frac{f(Y_j)g(Y_j)w_g(Y_i)}{p_g(Y_j)}$$

- Balance heuristic v.s. power heuristic

$$w_s(x)=\frac{n_s p_s(x)}{\sum_i n_i p_i(x)} \qquad w_s(x)=\frac{(n_s p_s(x))^\beta}{\sum_i (n_i p_i(x))^\beta}$$

Multiple importance sampling

- Assume a sample X is drawn from p_f where $p_f(X)$ is small, thus $f(X)$ is small if p_f matches f. If, unfortunately, $g(X)$ is large, then standard importance sampling gives a large value $\frac{f(X)g(X)}{p_f(X)}$
- However, with the balance heuristic, the contribution of X will be

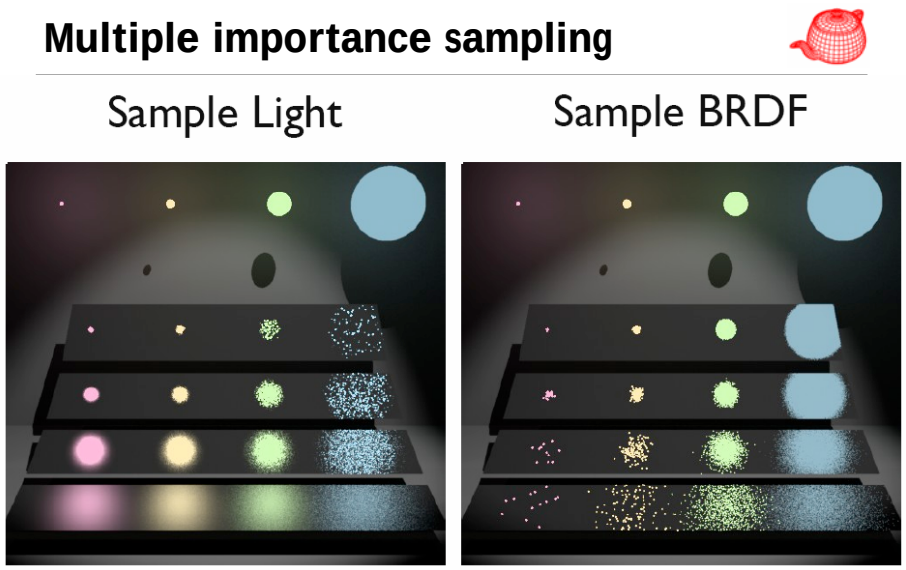
$$\frac{f(X)g(X)w_f(X)}{p_f(X)}=\frac{f(X)g(X)}{p_f(X)}\frac{n_f p_f(X)}{n_f p_f(X)+n_g p_g(X)}$$
$$=\frac{f(X)g(X)n_f}{n_f p_f(X)+n_g p_g(X)}$$

Multiple importance sampling

$$L_o(p,\omega_o)=\int_\Omega f_r(p,\omega_o,\omega_i)L_d(p,\omega_i)|\cos\theta_i|\,d\omega_i$$

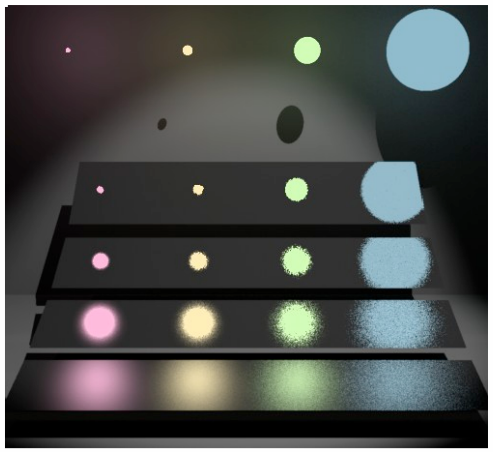
- If we sample based on either L or f, it often performs poorly.
- Consider a near-mirror BRDF illuminated by an area light where L's distribution is used to draw samples. (It is better to sample by f.)
- Consider a diffuse BRDF and a small light source. If we sample according to f, it will lead to a larger variance than sampling by L.
- It does not work by averaging two together since variance is additive.

Multiple importance sampling



Multiple importance sampling

Result: better than either of the two strategies alone



Monte Carlo for rendering equation

$$L_o(\mathrm{p},\omega_o) = L_e(\mathrm{p},\omega_o) + \int_\Omega f(\mathrm{p},\omega_o,\omega_i) L_i(\mathrm{p},\omega_i) |\cos\theta_i| d\omega_i$$

- Importance sampling: sample ω_i according to BxDF f and L (especially for light sources)
- If don't know anything about f and L, then use cosine-weighted sampling of hemisphere to find a sampled ω_i

Sampling reflection functions

```
Spectrum BxDF::Sample_f(const Vector &wo,
Vector *wi, float u1, float u2, float *pdf){
  *wi = CosineSampleHemisphere(u1, u2);
  if (wo.z < 0.) wi->z *= -1.f;
  *pdf = Pdf(wo, *wi);
  return f(wo, *wi);
}
```

For those who modified **Sample_f**, **Pdf** must be changed accordingly

```
float BxDF::Pdf(Vector &wo, Vector &wi) {
  return SameHemisphere(wo, wi) ?
          fabsf(wi.z) * INV_PI : 0.f;
}
```

Pdf() *is useful for multiple importance sampling.*

Sampling microfacet model

- Microfacet distribution D
- Fresnel reflection F
- Geometric attenuation G

$$f_r(\omega_i\omega_o) = \frac{D(\omega_h)\,G(\omega_i,\omega_o)\,F(\omega_i,\omega_h)}{4\cos\theta_i\cos\theta_o}$$

Too complicated to sample according to f, sample D instead. It is often effective since D accounts for most variation for f.

Sampling microfacet model

```
Spectrum Microfacet::Sample_f(const Vector &wo,
  Vector *wi, float u1, float u2, float *pdf) {
  distribution->Sample_f(wo, wi, u1, u2, pdf);
  if (!SameHemisphere(wo, *wi))
  return Spectrum(0.f);
  return f(wo, *wi);
}
float Microfacet::Pdf(const Vector &wo,
      const Vector &wi) const {
  if (!SameHemisphere(wo, wi)) return 0.f;
  return distribution->Pdf(wo, wi);
}
```

Sampling Blinn microfacet model

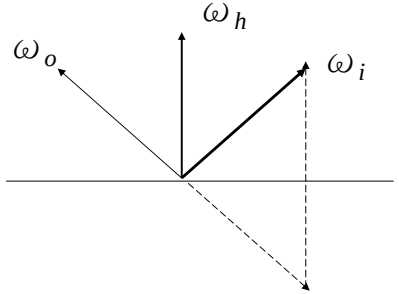
- Blinn distribution: $D(\cos\theta_h) = \frac{e+2}{2\pi}(\cos\theta_h)^e$
- Generate ω_h according to the above function

$$\phi_h = 2\pi\xi_2$$

$$\cos\theta_h = \sqrt[e+1]{\xi_1}$$

- Convert ω_h to ω_i

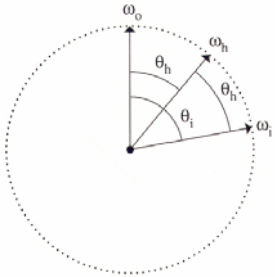
$$\omega_i = -\omega_o + 2(\omega_o\cdot\omega_h)\omega_h$$



Sampling Blinn microfacet model

- Convert half-angle PDF to incoming-angle PDF, that is, change from a density in term of ω_h to one in terms of ω_i

$\theta_i = 2\theta_h$ and $\phi_i = \phi_h$



transformation method

$$p_y(y) = \left(\frac{dy}{dx}\right)^{-1} p_x(x)$$

$$d\omega_i = \sin\theta_i\, d\theta_i\, d\phi_i$$

$$d\omega_h = \sin\theta_h\, d\theta_h\, d\phi_h$$

$$\frac{d\omega_h}{d\omega_i} = \frac{\sin\theta_h\, d\theta_h\, d\phi_h}{\sin\theta_i\, d\theta_i\, d\phi_i} = \frac{\sin\theta_h\, d\theta_h\, d\phi_h}{\sin 2\theta_h\, 2 d\theta_h\, d\phi_h} = \frac{\sin\theta_h}{4\cos\theta_h\sin\theta_h}$$

$$= \frac{1}{4\cos\theta_h} \rightarrow p(\theta) = \frac{p_h(\theta)}{4(\omega_o\cdot\omega_h)}$$

Sampling anisotropic microfacet model

- Anisotropic model (after Ashikhmin and Shirley) for the first quadrant of the unit hemisphere

$$D(\omega_h) = \sqrt{(e_x+1)(e_y+1)}(\omega_h \cdot n)^{e_x \cos^2\phi + e_y \sin^2\phi}$$

$$\phi = \arctan\left(\sqrt{\frac{e_x+1}{e_y+1}}\tan\left(\frac{\pi\xi_1}{2}\right)\right)$$

$$\cos\theta_h = \xi_2^{\left(e_x \cos^2\phi + e_y \sin^2\phi + 1\right)^{-1}}$$

Estimate reflectance

$$\rho_{hd}(\omega_o) = \int_\Omega f_r(\omega_o,\omega_i)\,|\cos\theta_i|\,d\omega_i$$

$$\frac{1}{N}\sum_{i=1}^{N}\frac{f_r(\omega_o,\omega_i)\,|\cos\theta_i|}{p(\omega_i)}$$

```
Spectrum BxDF::rho(Vector &w, int nS, float *S)
{
  if (!S) {
    S=(float *)alloca(2*nS*sizeof(float));
    LatinHypercube(S, nS, 2);
  }
  Spectrum r = 0.;
  for (int i = 0; i < nS; ++i) {
    Vector wi;
    float pdf = 0.f;
    Spectrum f=Sample_f(w,&wi,S[2*i],S[2*i+1],&pdf);
    if (pdf > 0.) r += f * fabsf(wi.z) / pdf;
  }
  return r / nS;
}
```

Estimate reflectance

$$\rho_{hh} = \frac{1}{\pi}\int_\Omega\int_\Omega f_r(\omega_o,\omega_i)\,|\cos\theta_i\cos\theta_o|\,d\omega_i d\omega_o$$

$$\frac{1}{\pi}\frac{1}{N}\sum_{i=1}^{N}\frac{f_r(\omega_i',\omega_i'')\,|\cos\theta_i'\cos\theta_i''|}{p(\omega_i')\,p(\omega_i'')}$$

```
Spectrum BxDF::rho(int nS, float *S) const
{
  if (!S) {
    S = (float *)alloca(4 * nS * sizeof(float));
    LatinHypercube(S, nS, 4);
  }
  Spectrum r = 0.;
  for (int i = 0; i < nS; ++i) {
    Vector wo, wi;
    wo = UniformSampleHemisphere(S[4*i], S[4*i+1]);
    float pdf_o = INV_TWOPI, pdf_i = 0.f;
    Spectrum f
         =Sample_f(wo,&wi,S[4*i+2],S[4*i+3],&pdf_i);
    if (pdf_i > 0.)
      r += f * fabsf(wi.z * wo.z) / (pdf_o * pdf_i);
  }
  return r / (M_PI*nS);
}
```

Sampling BSDF (mixture of BxDFs)

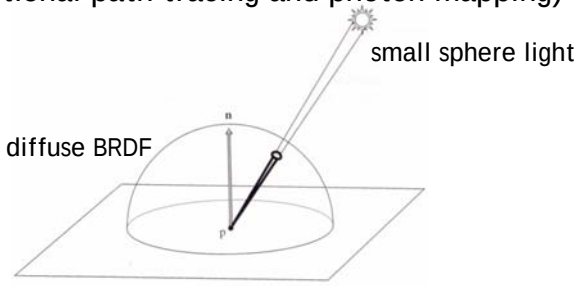
- Sample from the density that is the sum of individual densities

$$p(\omega) = \frac{1}{N}\sum_{i=1}^{N} p_i(\omega)$$

- Three uniform random numbers are used, the first one determines which BxDF to be sampled and then sample that BxDF using the other two random numbers

Sampling light sources

- Direct illumination from light sources makes an important contribution, so it is crucial to be able to generates
  - Sp: samples directions from a point p to the light
  - Sr: random rays from the light source (for bidirectional light transport algorithms such as bidirectional path tracing and photon mapping)



Interface

```
virtual Spectrum Sample_L(const Point &p,
  float u1, float u2, Vector *wi, float *pdf,
  VisibilityTester *vis) const = 0;
virtual float Pdf(const Point &p,
  const Vector &wi) const = 0;
```

We don't have normals for volume.

```
virtual Spectrum Sample_L(… Normal &n, …) {
  return Sample_L(p, u1, u2, wi, pdf, vis);
}
```

If we know normal, we could add consine falloff to better sample L.

```
virtual float Pdf(… Normal &n, …) {
    return Pdf(p, wi);
}
```

Default on simply forward to the one without normal.

```
virtual Spectrum Sample_L(const Scene *scene,
  float u1, float u2, float u3, float u4,
  Ray *ray, float *pdf) const = 0;
```

Rays leaving lights

Point lights

- Sp: delta distribution, treat similar to specular BxDF
- Sr: sampling of a uniform sphere

Point lights

```
Spectrum Sample_L(const Point &p, float u1, float u2,
  Vector *wi, float *pdf, VisibilityTester *vis)
{
  *pdf = 1.f;
  return Sample_L(p, wi, visibility);
}
```

delta function

```
float Pdf(Point &, Vector &) const
{
  return 0.;
}
```

for almost any direction, pdf is 0

```
Spectrum Sample_L(Scene *scene, float u1, float u2,
  float u3, float u4, Ray *ray, float *pdf) const
{
  ray->o = lightPos;
  ray->d = UniformSampleSphere(u1, u2);
  *pdf = UniformSpherePdf();
  return Intensity;
}
```

Spotlights

- Sp: the same as a point light
- Sr: sampling of a cone (ignore the falloff)

$$p(\omega) = c \text{ over cone} \longrightarrow p(\theta,\phi) = c\sin\theta \text{ over } [0,\theta_{\max}]\times[0,2\pi]$$

$$1 = \int_{\phi=0}^{2\pi}\int_{\theta=0}^{\theta'} c\sin\theta d\theta d\phi = 2\pi c(1-\cos\theta_{\max}) \longrightarrow p(\theta,\phi) = \frac{\sin\theta}{2\pi(1-\cos\theta_{\max})}$$

$$p(\theta) = \int_{\phi=0}^{2\pi} \frac{\sin\theta}{2\pi(1-\cos\theta_{\max})} d\phi = \frac{\sin\theta}{1-\cos\theta_{\max}}$$

$$P(\theta) = \int_{\theta=0}^{\theta'} \frac{\sin\theta}{1-\cos\theta_{\max}} d\theta = \frac{1-\cos\theta'}{1-\cos\theta_{\max}} = \xi_1 \longrightarrow \cos\theta = (1-\xi_1) + \xi_1\cos\theta_{\max}$$

$$p(\phi\,|\,\theta) = \frac{p(\theta,\phi)}{p(\theta)} = \frac{1}{2\pi} \longrightarrow P(\phi'|\,\theta) = \int_{\phi=0}^{\phi'} \frac{1}{2\pi} d\phi = \frac{\phi'}{2\pi} = \xi_2 \longrightarrow \phi = 2\pi\xi_2$$

Spotlights

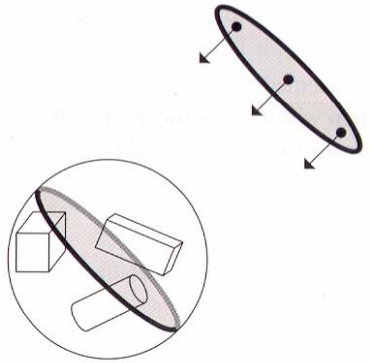
```
Spectrum Sample_L(Point &p, float u1, float u2,
  Vector *wi, float *pdf, VisibilityTester *vis)
{
   *pdf = 1.f;
   return Sample_L(p, wi, visibility);
}
float Pdf(const Point &, const Vector &)
{  return 0.; }

Spectrum Sample_L(Scene *scene, float u1, float u2,
  float u3, float u4, Ray *ray, float *pdf)
{
  ray->o = lightPos;
  Vector v = UniformSampleCone(u1, u2,cosTotalWidth);
  ray->d = LightToWorld(v);
  *pdf = UniformConePdf(cosTotalWidth);
  return Intensity * Falloff(ray->d);
}
```

Projection lights and goniophotometric lights

- Ignore spatial variance; sampling routines are essentially the same as spot lights and point lights

Directional lights

- Sp: like point lights
- Sr: create a virtual disk of the same radius as scene's bounding sphere and then sample the disk uniformly.



Directional lights

```
Spectrum Sample_L(Scene *scene, float u1, float u2,
  float u3, float u4, Ray *ray, float *pdf) const
{
  Point worldCenter;
  float worldRadius;
  scene->WorldBound().BoundingSphere(&worldCenter,
                                     &worldRadius);
  Vector v1, v2;
  CoordinateSystem(lightDir, &v1, &v2);
  float d1, d2;
  ConcentricSampleDisk(u1, u2, &d1, &d2);
  Point Pdisk =
      worldCenter + worldRadius * (d1 * v1 + d2 *
  v2);
  ray->o = Pdisk + worldRadius * lightDir;
  ray->d = -lightDir;
  *pdf = 1.f / (M_PI * worldRadius * worldRadius);
  return L;
}
```

Area lights

- Defined by shapes
- Add shape sampling functions for **Shape**
- Sp: uses a density with respect to solid angle from the point p

  `Point Shape::Sample(Point &P, float u1, float u2, Normal *Ns)`
- Sr: generates points on the shape according to a density with respect to surface area

  `Point Shape::Sample(float u1, float u2, Normal *Ns)`
- `virtual float Shape::Pdf(const Point &Pshape) { return 1.f / Area(); }`

Area light sampling method

- Most of work is done by **Shape**.

```
Spectrum Sample_L(Point &p, Normal &n, float u1,
  float u2, Vector *wi, float *pdf,
  VisibilityTester *visibility) const {
  Normal ns;
  Point ps = shape->Sample(p, u1, u2, &ns);
  *wi = Normalize(ps - p);
  *pdf = shape->Pdf(p, *wi);
  visibility->SetSegment(p, ps);
  return L(ps, ns, -*wi);
}
float Pdf(Point &p, Normal &N, Vector &wi) const {
  return shape->Pdf(p, wi);
}
```

Area light sampling method

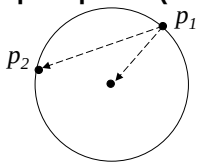
```
Spectrum Sample_L(Scene *scene, float u1, float u2,
  float u3, float u4, Ray *ray, float *pdf) const
{
  Normal ns;
  ray->o = shape->Sample(u1, u2, &ns);
  ray->d = UniformSampleSphere(u3, u4);
  if (Dot(ray->d, ns) < 0.) ray->d *= -1;
  *pdf = shape->Pdf(ray->o) * INV_TWOPI;
  return L(ray->o, ns, ray->d);
}
```

Infinite area lights

- Essentially an infinitely large sphere that surrounds the entire scene
- Sp:
  - normal given: cosine weighted sampling
  - otherwise: uniform spherical sampling
  - does not take directional radiance distribution into account
- Sr:
  - Uniformly sample two points p_1 and p_2 on the sphere
  - Use p_1 as the origin and p_2-p_1 as the direction
  - It can be shown that p_2-p_1 is uniformly distributed (Li *et. al.* 2003)

Infinite area lights

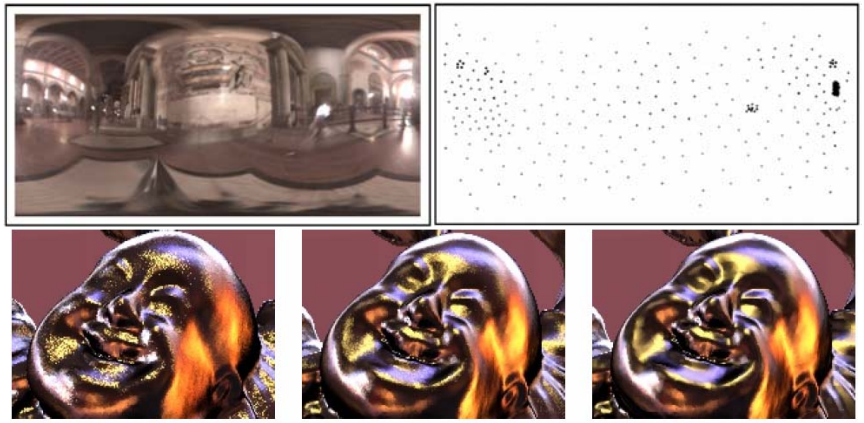
```
Spectrum Sample_L(Scene *scene, float u1, float u2,
  float u3, float u4, Ray *ray, float *pdf) const
{
  Point wC; float wR;
  scene->WorldBound().BoundingSphere(&wC, &wR);
  wR *= 1.01f;
  Point p1 = wC + wR * UniformSampleSphere(u1, u2);
  Point p2 = wC + wR * UniformSampleSphere(u3, u4);
  ray->o = p1;
  ray->d = Normalize(p2-p1);
```




```
  Vector to_center = Normalize(worldCenter - p1);
  float costheta = AbsDot(to_center,ray->d);
  *pdf = costheta / ((4.f * M_PI * wR * wR));
  return Le(RayDifferential(ray->o, -ray->d));
}
```

Sampling lights

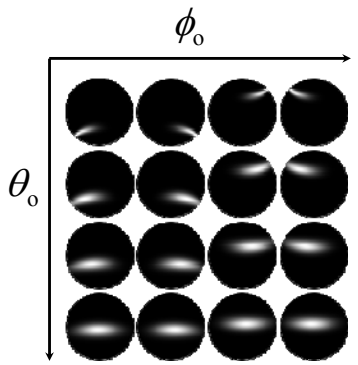
- Structured Importance Sampling of Environment Maps, SIGGRAPH 2003



Importance w/ 300 samples | Importance w/ 3000 samples | Structured importance w/ 300 samples

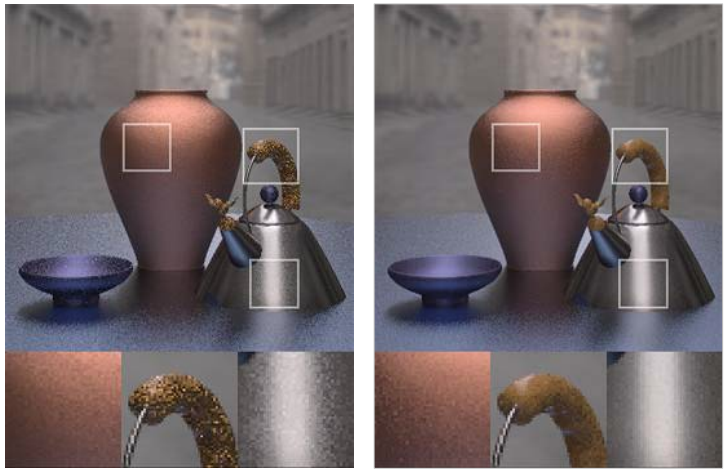
Sampling BRDF

- Efficient BRDF Importance Sampling Using A Factored Representation, SIGGRAPH 2004



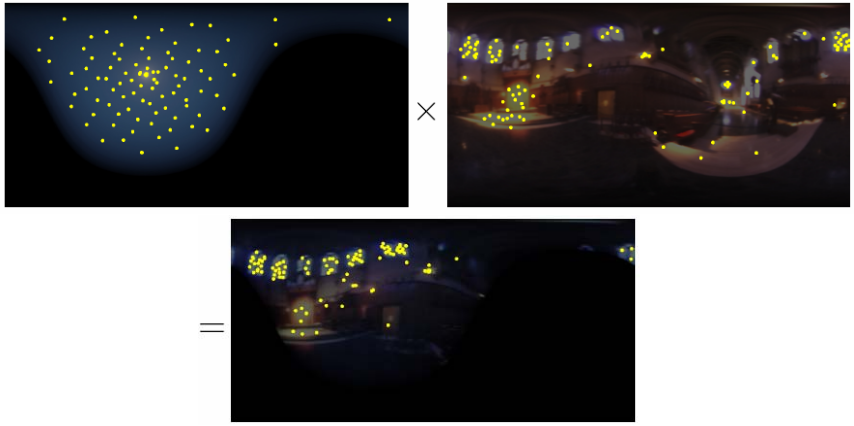
$$\approx \sum_{j=1}^{J} F_j(\omega_o) G_j(\omega_i)$$

Results (300 samples)

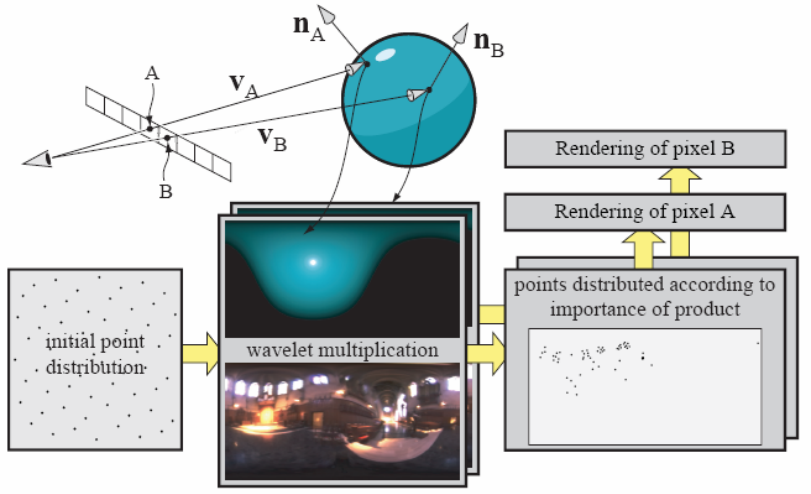


Sampling product

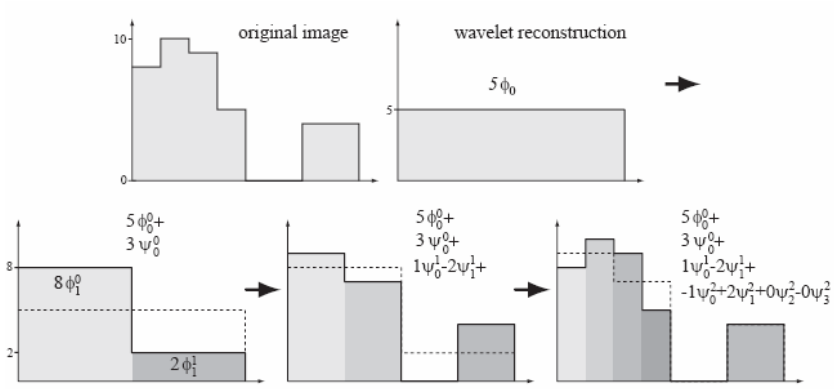
- Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions, SIGGRAPH 2005.



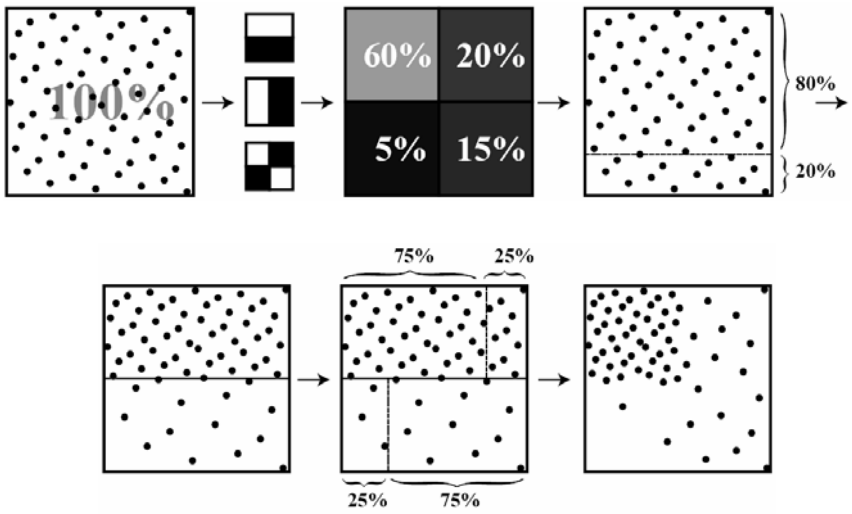
Sampling product



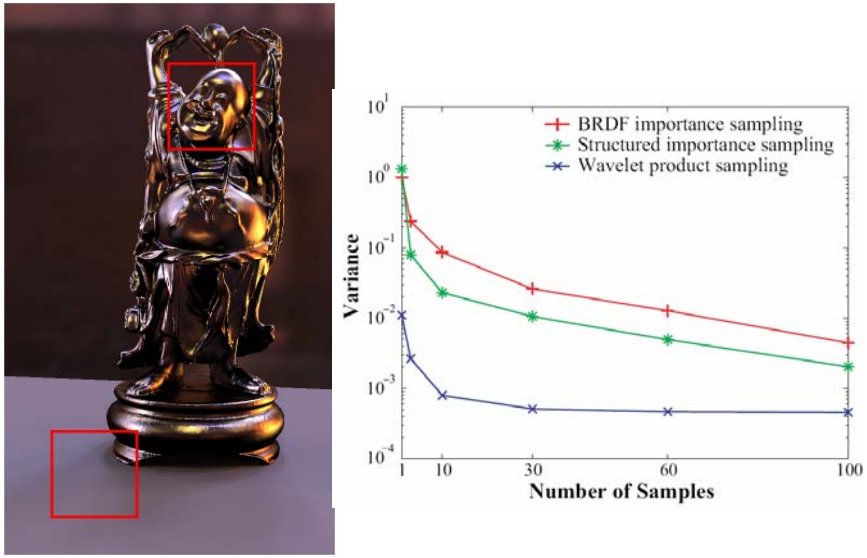
Wavelet decomposition



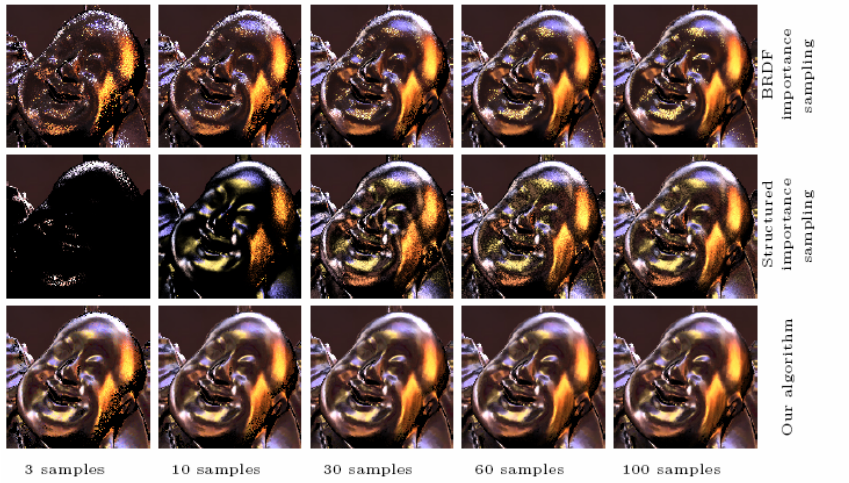
Sample warping



Results



Results



Results