

CS278 - Assignment 1

Monte Carlo

Given a description of a simple scene and camera, you are to write a program that will compute a, unit accurate, unbiased estimate, of the power arriving at each pixel in the sensor. To do this you will need to compute random samples of some appropriate integral. As the number of samples in your estimate increases, the variance of the estimator should decrease. Your submission will include the code you write, as well as images showing the result of your simulation using an increasing number of samples.

We will try to do an experiment similar to the one shown in Figure 2 of Veach.

1 Problem statement

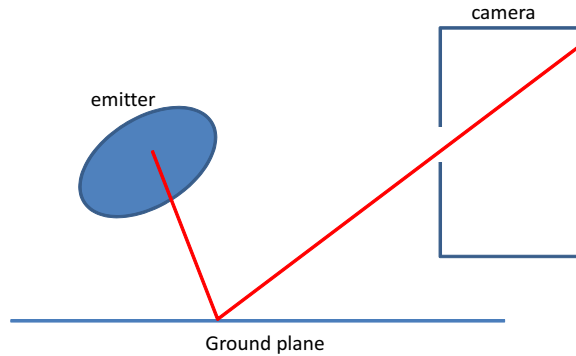


Figure 1: Our scene

The scene will be composed of:

- A circular diffuse light source that emits the outgoing radiance of 1 at every point, in every direction on its front side. The light source is a circular disk. Its center is the point $[?, ?, ?]^t$, normal of its supporting plane is $[0, 0, 1]^t$, and its radius will be a parameter r .
- A ground plane with BRDF $f = \frac{n+2}{2*\pi} \cos^n \alpha$, where α is the angle between the perfect bounce of the incoming direction and outgoing direction. n is a non negative integer, which will be an input parameter. The ground plane will be $y = ?$, (with normal $[0, 1, 0]^t$).
- A pin-hole camera consisting of a box like the one in the figure, and a film plane. The camera will point down the negative z axis. The center of the pin hole will be at $[0, 0, 0]^t$. The radius of the pin hole will be $0.001m$. The film has square-shape piece of a plane, is centered at the point $[0, 0, 2]^t$, with normal $[0, 0, 1]^t$. The film has total height of 1m, and total width of

1m. It has 400x400 pixels arranged evenly (with a 1/2 pixel buffer on all sides). Each pixel measures incoming power in watts, with a simple box sensor function for each pixel (simply adds up all the incoming power to the pixel). We will only be interested in one-bounce light. That is: there is no direct contribution of power from the light source to the film, and there is no two-bounce light (i.e. the film plane and the light source reflect no light).

The input to the program is r , n and s : the number of samples per pixel.

The output of the program is a file with the values of the estimates for the incoming power at each of the 400x400 pixels in the image. You can output these simply as ASCII floats in row-major order.

In all of the integration steps, we will only sample one point in a pixel (at its center), and one point over the aperture (at its center). This will determine a single point on the ground plane. We will be using multiple samples over the area of the emitter (alt over the incoming directions at the point on the ground plane).

2 Part I

In this part we will use uniform area sampling over the emitting disk.

For each pixel ij , start by writing its integral in the form

$$\int_{S_1} \int_{S_2} \int_{pix} dA_1 dA_2 dA_3 \dots$$

Where S_1 is the emmitter, and S_2 is the ground plane.

Then put it in the form

$$\int_{S_1} \int_{ap} \int_{pix} dA_1 dw_{i23} dA_3 \dots$$

Since we assume that pixel and the aperture is very small, we can assume the integrand is constant over that variation and reduce our integrand to the form

$$\dots \int_{S_1} dA_1 \dots$$

It will take a bit of thinking to get the leading constant correct. In particular, you will need to compute the solid angle covered by the apurture above \tilde{x}_3 .

Thus we shoot a ray from the center of pixel ij to S_2 and then integrate over the incoming rays at that surface using Monte Carlo.

To check the units, we can choose a diffuse surface $n = 0$, a very small r , and set $s = 1$.

3 Part II

In this part you will compute exactly the same estimate, but use imporance sampling of the incoming directions based on the BRDF.

For each pixel ij , again, start by writing its integral in the form

$$\int_{S_1} \int_{S_2} \int_{pix} dA_1 dA_2 dA_3 \dots$$

Where S_1 is the emmitter, and S_2 is the ground plane.

Then put it in the form

$$\int_{S_1} \int_{ap} \int_{pix} dw_{i12} dw_{i23} dA_3 \dots$$

Then use the assumption of small pixel and aperture to reduce the integrand to the form

$$\dots \int_{S_1} dw_{i12} \dots$$

4 Output format

The output of the program is simply a (ASCII) text file with the width and height of the picture, and float values of the power arriving at each of the 400x400 pixels, listed in row-major order, as in:

```
<width (W) in pixels of image>
<height (H) in pixels of image>
<WxH floats>
```

5 Help

5.1 Viewing

A simple program (asstlview.c) is provided that converts your output to a ppm file. This program also prints to the console the maximum power arriving at a pixel, which it uses as normalizing constant to produce the image. You may also call the program providing your own normalizing constant.

You can use this program to check if the output from parts I and II are approximately the same, by running it on both while passing the same normalizing constant.

5.2 Sampling

In order to generate a uniform sample on a disk, you may make use of the following routines which computes a random point on a unit circle using rejection sampling:

```
void sampleUnitDisk(float& x, float& y)
{
    do { x = 2*drand48()-1; y = 2*drand48()-1; } while (x*x+y*y > 1);
}
```

Note that `drand48` is defined in UNIX under `cstdlib`, but is not available on windows. On windows you may need to use `((long double)rand())/((long double)RAND_MAX)` instead.

To do directional importance sampling, you should use the PDF $p(w_{i12}) = \frac{n+1}{2\pi} \cos^n \alpha$. This can be done as explained on page 9 of Lafortune. That page explains how to do the sampling over the upper hemisphere with central vector $[0, 1, 0]^t$, where α is the angle with the central vector. You will need to rotate the resulting sample so that the central ray is the “bounce ray” $B(-w_{o23})$.