UNIVERSITAT DE BARCELONA
MACHINE LEARNING IN COMPUTER GRAPHICS

Assignment 3,4 Report

Xiao Fei

February 2017

In this report, I will show the result of Assignment 3 and 4 with explaining the implementation and the discussing the results.

1 ASSIGNMENT 3 BAYESIAN MONTE-CARLO

1.1 Implementation

We complete the compute_inv_Q() function which is to compute the Q matrix. The core of the function is using the given covariance function to calculate the distance between each pair of sample points, i.e. sample point i and j.

```
Q[i][j] = self.cov_func.eval(self.samples_pos[i], self.samples_pos[j])
```

The core of the function compute_z() is shown in following lines. It is based on the formula $z = (\int k(w_1, w)\,dw, ..., k(w_n, w)\,dw)$.

```
for sample,prob in zip(sample_set_z,probab):
    sum_integral+= self.cov_func.eval(omega_i,sample)*self.p_func.eval(sample) / prob
```

The compute_BMC() function is rather straightforward which sum the multiply result of weights and sample value for each sample position.

```
for (w,y) in zip(self.weights,self.samples_val):
    res +=  y * w
```

1.2 Benchmark

The Figure1.1 shows that the Bayesian Monte-Carlo estimate the ground truth much better than the classic Monte-Carlo. With more samples, the classic Monte-Carlo reduce the error by almost a half. With same amount increasing of sample rays, the Bayesian Monte-Carlo reduce slightly, and even increase the error after 80 samples.

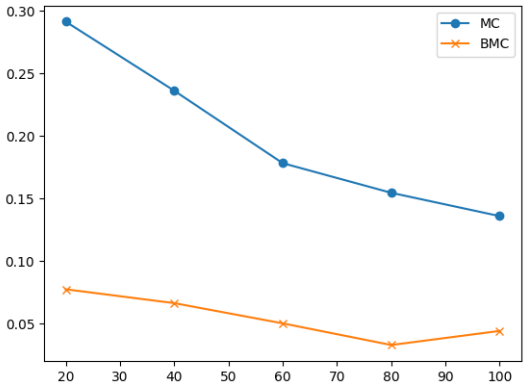
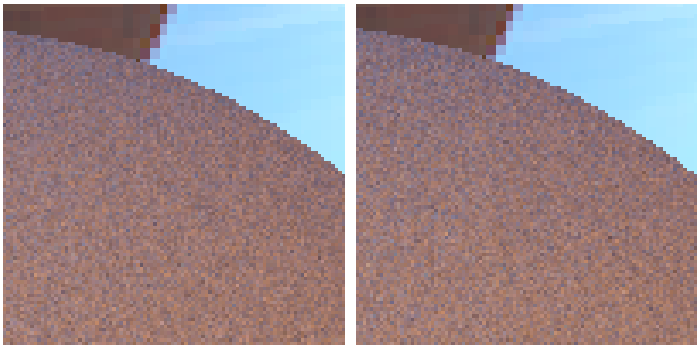


Figure 1.1: Benchmark: Bayesian Monte-Carlo vs Classic Monte-Carlo

1.3 Rendering

The rendering result of using Bayesian Monte-Carlo have shown a similar image with Classic Monte-Carlo method. From a closer inspection, we noticed that Bayesian Monte-Carlo do produce a less noise rendering image compared to the classic Monte-Carlo.



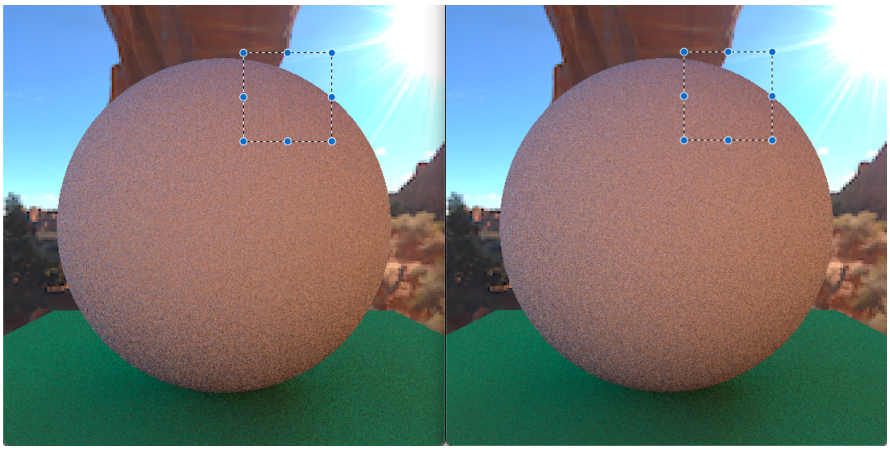


Figure 1.2: Aligned Comparison CMC vs BMC

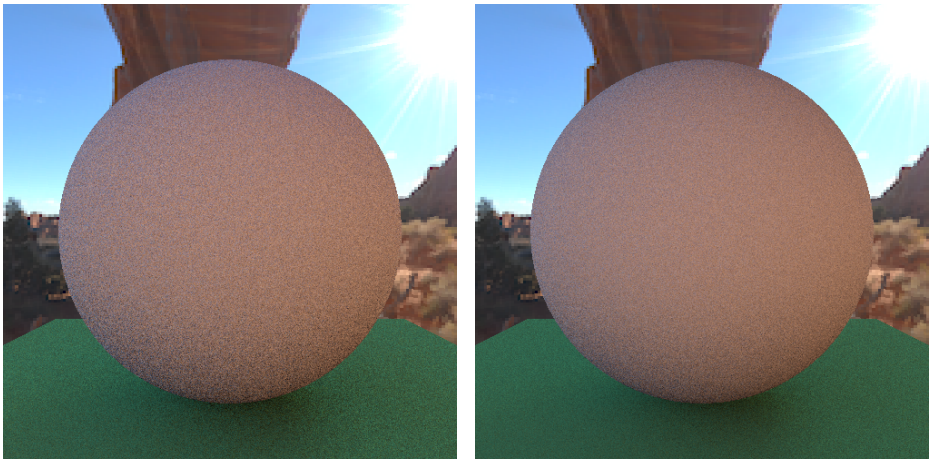


Figure 1.3: The comparison of different number of samples, 40(left), 80(right)

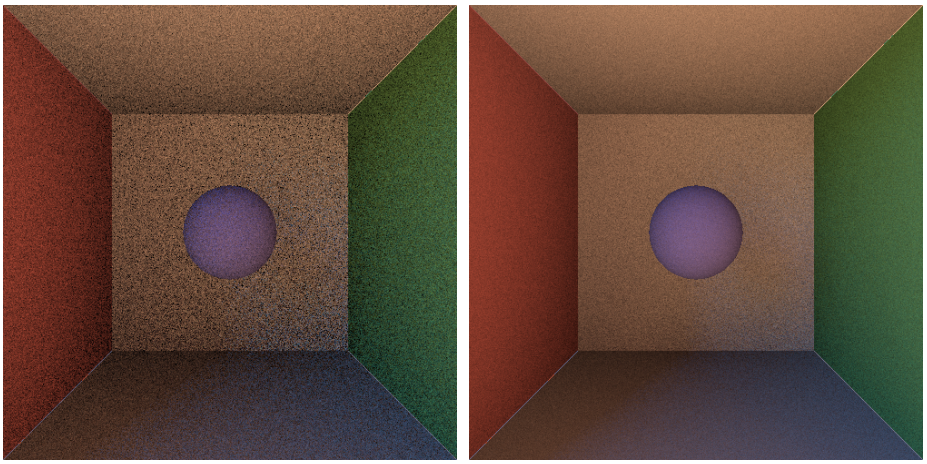


Figure 1.4: The comparison of number of samples in Cornell scene

2 Assignment 4 Importance Sampling

2.1 Classic Monte-Carlo Importance Sampling

We implement the importance sampling by simply replace the UniformPDF to CosineLobe(1) inside the CMCIntegrator.

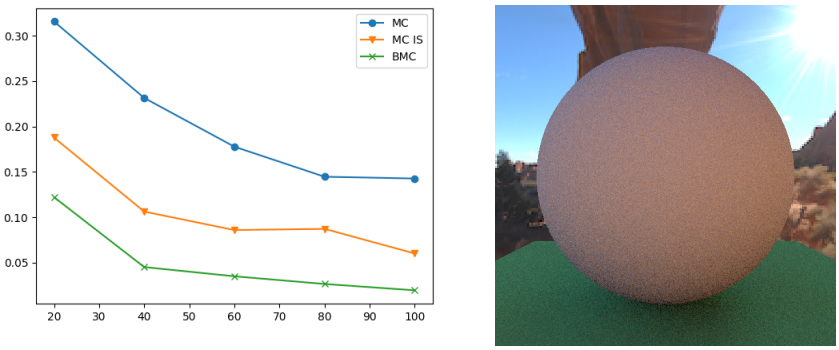


Figure 2.1: Benchmark and Rendering

2.2 Bayesian Monte-Carlo Importance Sampling

We implemented the Bayesian Monte-Carlo Importance Sampling similar to Bayesian Monte-Carlo but using CosineLobe(1) instead of Constant(1). Another changes is the CosinePDF used in the sample_set_hemisphere function. We set the running time to 10 and averaging the results from these 10 runs.

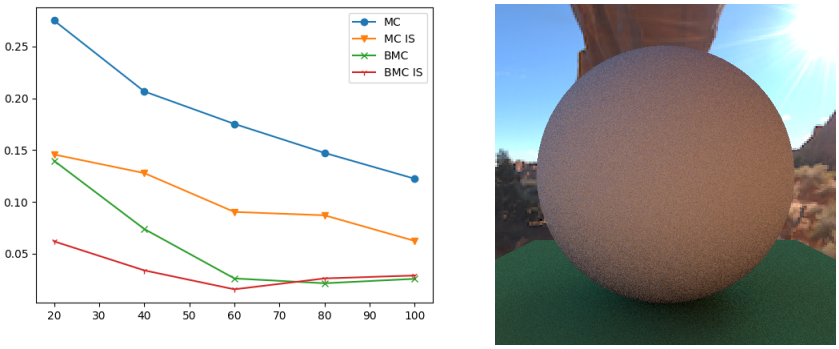


Figure 2.2: Benchmark and Rendering