Primary Sample Space Metropolis Light Transport with Delayed Rejection

YUAN MENG, Carnegie Mellon University, USA

In the final project, I explored PSSMLT and the delayed rejection technique which is applied on top of PSSMLT. I also make DIRT able to sample time so that I can create motion-blur pictures.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Metropolis Light Transport

1 ABOUT FINAL PROJECT

This is the final project report of CMU 15-668 Physics-Based Rendering course. I've implemented in DIRT the below features:

- A PSSMLT integrator
- A delayed rejection improvement on PSSMLT
- The ability to sample time

2 INTRODUCTION

PSSMLT[1](Primary Sample Space Metropolis Light Transport) is first proposed by Kelemen as a easier version of the original Metropolis Light Transport[3] in path space. These method are all based on Bidirectional Path Tracing but in DIRT we do not have a BDPT. Although it was originally designed for BDPT, the idea of using primary sample space also applies to any path tracer who utilizes random numbers to generate paths. This is the reason why I could integrate a PSSMLT into DIRT's unidirectional path tracers. Delayed rejection[2] is an improvement on the mutation/perturbation part of the PSSMLT algorithm, it will explore the primary sample space with two stage mutations, which gives PSSMLT the ability to explore smaller steps. Once we have a PSSMLT integrator, we could use this to sample time. It will naturally focus more on the part where the luminance is high, which is a good way to sample time.

3 PRELIMINARIES

3.1 Metropolis Light Transport

The original MLT is proposed by Veach, based on BDPT. The algorithm focuses in path space as all mutations and perturbations are made in path space. The complexity of path space mutations makes the original MLT very hard to implement.

3.2 Primary Sample Space MLT

Later, Kelemen proposed a simple mutation strategy for MLT, which is PSSMLT. The core of PSSMLT is making mutations only in Primary Sample Space - an infinite dimension space of all the random numbers used in path generation.

The idea is very simple, if the random numbers used to generate one path are very close to the numbers used to generate another. These two paths are of high probability to be similar and thus have the probability to maintain high contribution to the image. The mutation strategy of PSSMLT generally has two parts: first is the small local step aims to explore high peak of the contribution, second is the large global step to step out of local maximum and explore other high contribution regions.

One great property of PSSMLT is that, if the mutation of random number is symmetric, for example, a normal distribution, the transition probability between two states can be cancelled out. PSSMLT is a lot easier to implement than the original one.

3.3 Delayed Rejection

Delayed rejection is used in PSSMLT to help it better explore locally. When a local mutation is made but get rejected, delayed rejection technique will re-try to make a smaller mutation, hoping this smaller mutation will generate a path closer to the original one. The idea is very simple but this technique requires more computation on transition probabilities. A spacial designed mutation is needed to make these probabilities cancel out.

3.3.1 Circular distribution to cancel out transition probability. Denote current state x, first stage proposal state y, second stage proposal z. Q_1 and Q_2 denotes the transition probability of the first and second mutation. The second acceptance rate is

$$a_2(x,z) = min(1, \frac{f(z)Q_1(y|z)Q_2(x|y,z)[1-a_1(z,y)]}{f(x)Q_1(y|x)Q_2(z|y,x)[1-a_1(x,y)]}$$

Note that the transition probability $Q_1(y|z)$ and $Q_1(y_x)$ are generally not the same as z is generated from x using Q_2. However, if we let

$$\|z-y\| = \|x-y\|$$

Then these two probability will be the same. This is the logic behind using a circular distribution such as Wrapped Cauchy and von Mises.

3.4 Sampling Time

It is natural to use MLT algorithm to sample time robustly. All we need to do is make the renderer able to compute the location of the scene objects at certain time.

4 DESIGN AND IMPLEMENTATION

4.1 PSSMLT

PSSMLT contains two part, a PSS sampler and a modified main rendering loop to do acceptance and rejection.

4.1.1 PSS Sampler. A sampler is nothing else but a class which can return random numbers to be used in path generation process. We already have many kinds of samplers from the simplest uniform random number generator to Halton sampler. PSS sampler will keep two arrays, one is current random number sequence, another is proposed random number sequence. The proposed sequence is obtained from mutating every element in the random number sequence. Once a routine requires a random number from the PSS sampler, it will return the number in proposed array in sequence. PSS sampler should also support accept/reject operations in order

Author's address: Yuan Meng, yuanm2@andrew.cmu.edu, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, Pennsylvania, USA, 15213.

to maintain the current random number sequence. For example, once accepted, it should let the proposed sequence be the current sequence and clear any metadata to make the sampler prepared for the next iteration. A lazy evaluation framework is used, as the original paper, to save the memory and time consumption of unnecessary random number generation.

Mutation. For local step, I use a normal distribution. For large step, I just uniformly sample from $[0, 1)$. These two mutations ensure that the transition probability is always the same.

Lazy Evaluation. Since a lazy evaluation of proposed sequence is used. We need to make sure that we make up for the mutations we did not operate. Remember that the Primary Sample Space is of infinite dimension. One round of mutation theoretically should mutates an infinite number of random numbers. This is not practical so in the implementation the number is only mutated when it is used. Let's assume the last large step happened at iteration N, now is iteration M and it is the first time we use random number $XProposed[i]$, we should first generate $XProposed[i]$ uniformly in $[0, 1)$ and do local mutations for $M - N$ times.

4.1.2 *Main rendering Loop.*

Normalization Factor Computation(Burn in). The burn in stage of PSSMLT is also the normalization factor computation stage. Here we use different random seeds to estimate the contribution of the path generated by each seed. Note that here the random number sequence of the first iteration of the generation must be large step(uniform $[0, 1)$ float). After this step. We have some random seeds which generate high contribution initially and the normalization factor.

MLTRender. Due to the structure of DIRT, I have to write another render function as MLT does not use stratified pixels.

Main Loop. We choose those good seeds, build PSS Sampler using these seeds and start PSSMLT. We don't need to compute the transition probability as it all cancels out.

4.2 Delayed Rejection

4.2.1 *Delayed Rejection Sampler.*

This sampler is very similar to PSSMLT but it will maintain three arrays: current sequence, first proposal sequence and second proposal sequence. We need to add some bool flag to the function calls to indicate which stage we are in.

Mutation. The first mutation is Kelemen's mutation in its original PSSMLT. This mutation is somewhat aggressive(more towards a global exploration). The second one is a wrapped Cauchy distribution, which is a timid one which limits the exploration range. By combining these two different kinds of mutation. We can achieve a better local exploration.

Main Loop. The main loop is just has one more proposal involved. Now we make three contributions to the picture in one iteration. Everything else will be the same as PSSMLt.

4.3 Sampling time

4.3.1 *Decomposition of Transformation Matrix.*

A transformation matrix is

$$M = T \cdot R \cdot S$$

where T is the translation matrix and R is the rotation matrix and S is the scaling matrix. The order of these three matrix can be exchanged. Directly interpolating the transformation matrix will not create a good intermediate state. So separately interpolating these three matrix will be better. It is better that we use Quarternion to express and interpolate rotation but I did not implement that.

4.3.2 *Interpolating Scene.*

In the current implementation, I have to create many scenes in advance and interpolating between these scenes. For a complicated scene this will take up huge amount of memory and it is not a good way. This is mainly because the BVH structure used in DIRT can not be easily modified, it may need to be reconstructed every time the object moves.

5 RESULTS AND EVALUATION

5.1 Correctness

I will evaluate the consistency of the two MLT integrators by comparing the results with a reference picture generated by unidirectional MIS integrator(2048 spp). Note that there are some black pixels in the MIS picture which I believe is caused by a NaN result in DIRT's MIS routine. But this won't affect the evaluation.

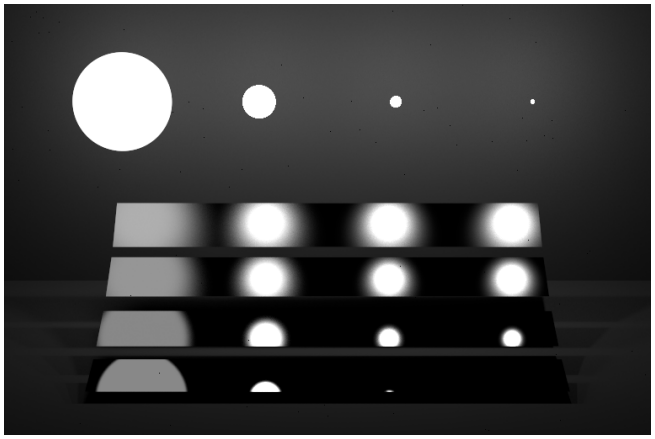

Fig. 1. Result from unidirectional MIS integrator 2048spp

5.1.1 *PSSMLT result.*

Path tracer is used as the internal integrator for the follwing pictures.

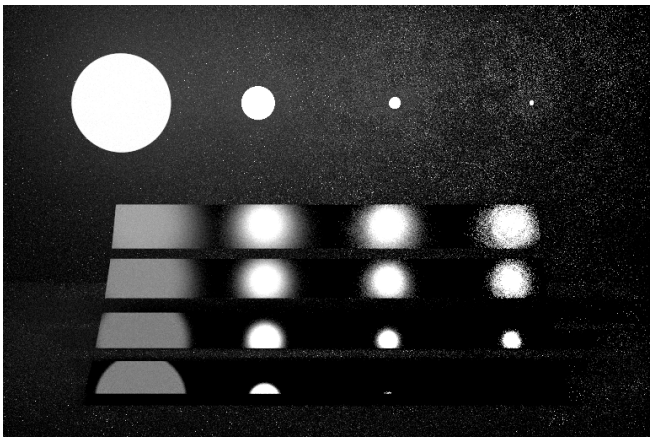

Fig. 2. Result from PSSMLT + Path Tracer 256spp

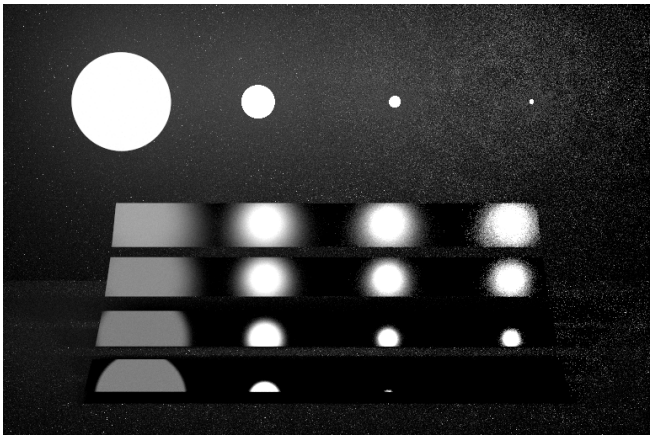

Fig. 3. Result from PSSMLT + Path Tracer 1024spp

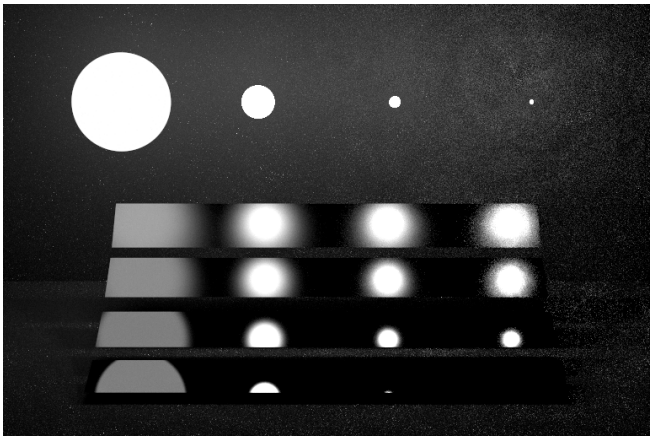

Fig. 4. Result from PSSMLT + Path Tracer 2048spp

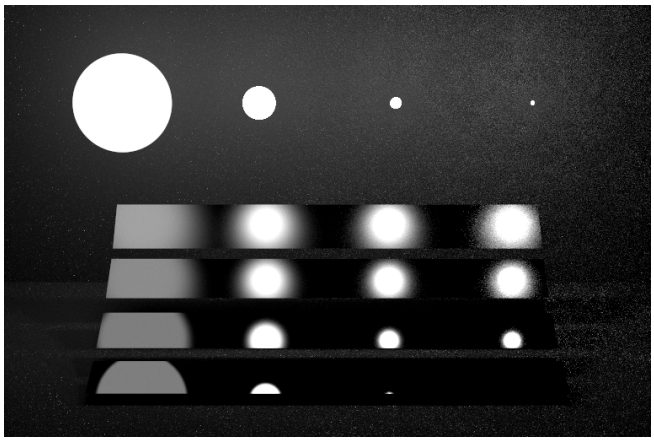

Fig. 5. Result from PSSMLT + Path Tracer 5000spp

Here is a result of 512SPP for PSSMLT + Path Tracer with MIS

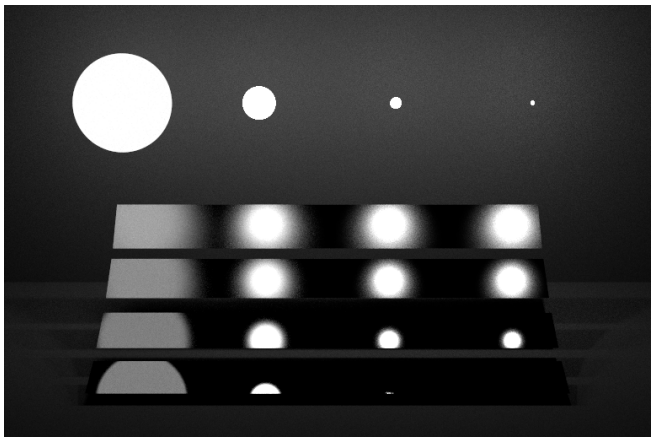

Fig. 6. Result from PSSMLT + MIS Path Tracer 512spp

5.1.2 DRPSSMLT result. Here are the results of different SPP for DRPSSMLT + Path Tracer

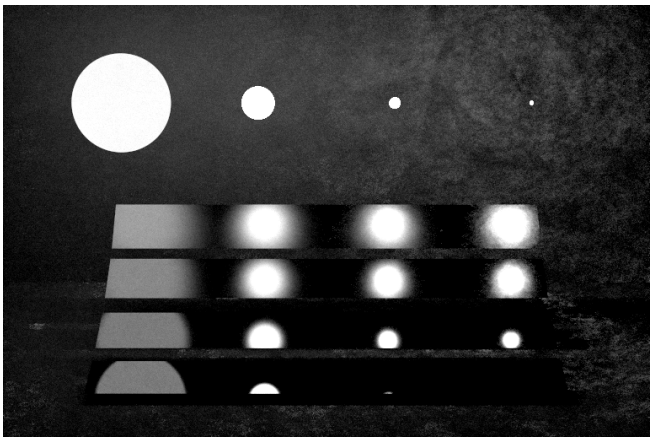

Fig. 7. Result from DRPSSMLT + Path Tracer 256spp

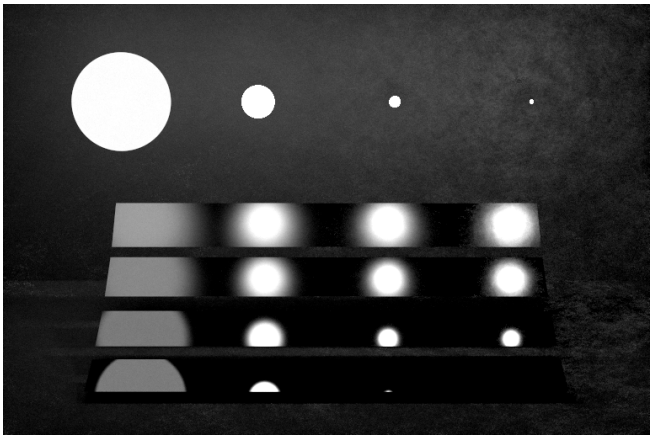

Fig. 8. Result from DRPSSMLT + Path Tracer 1024spp

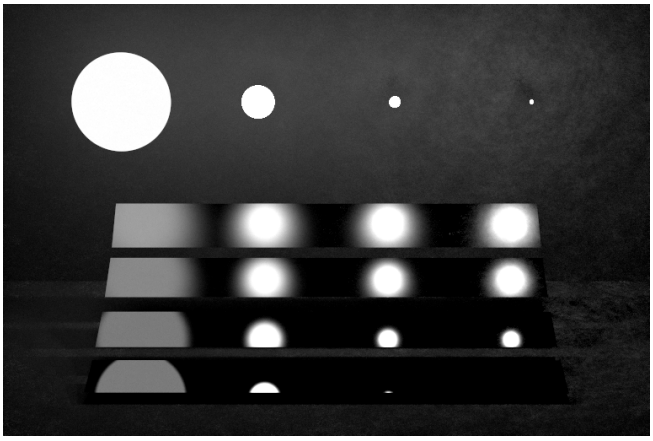

Fig. 9. Result from DRPSSMLT + Path Tracer 2048spp

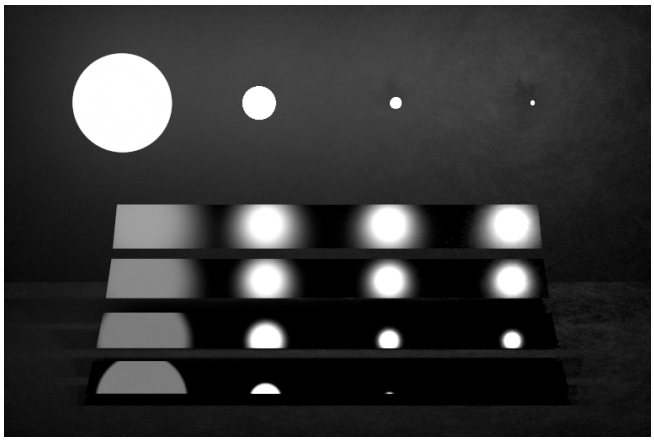

Fig. 10. Result from DRPSSMLT + Path Tracer 5000spp

Yannis pointed out during the presentation that the result of DRPSSMLT seems to have bias as the background seems to be structural, not noise. But As I increase the SPP numbers. It seems to be approaching the ref image. So I will think this could reach consistency

Here is a result of 512SPP for DRPSSMLT + Path Tracer with MIS

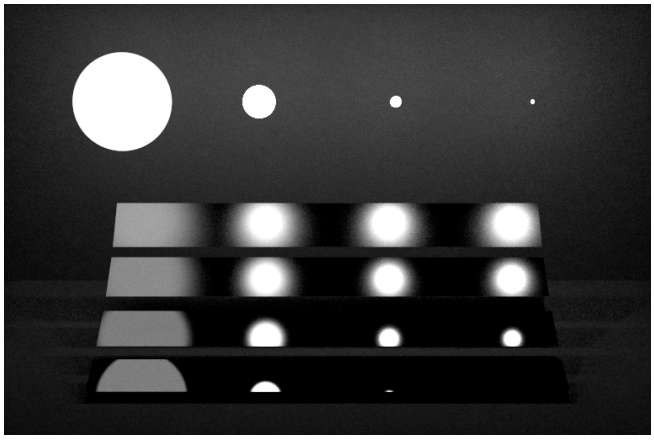

Fig. 11. Result from DRPSSMLT + MIS Path Tracer 512spp

5.1.3 Conclusion. Both PSSMLT and DRPSSMLT becomes closer to reference picture as spp increases. This may suggest they are impelemented correctly. The effect of applying PSSMLT/DRPSSMLT on top of an MIS(which is already very good in this scene) is not big.

5.2 Effectiveness

As I mentioned in the last section, applyting PSSMLT to MIS will not make big difference as MIS is good enough. So I will apply both PSSMLT and DRPSSMLT to path tracer, and compare them in the Veach scene.

First with 32SPP:

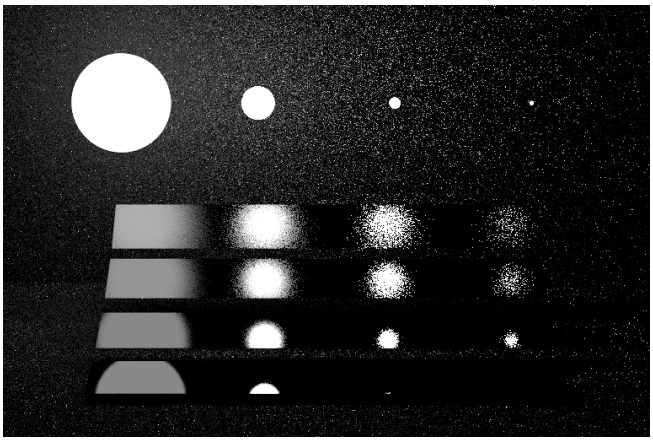

Fig. 12. Result from Path Tracer 32spp

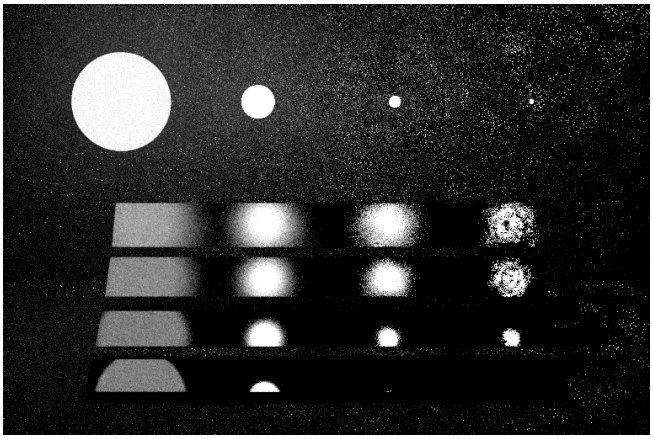

Fig. 13. Result from PSSMLT + Path Tracer 32spp $\sigma = 0.02$

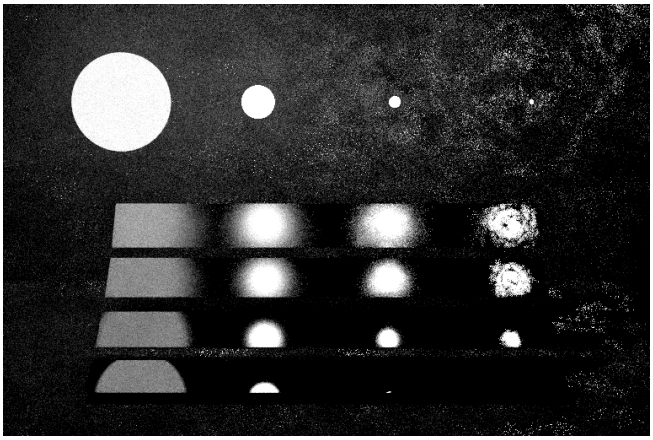

Fig. 14. Result from PSSMLT + Path Tracer 32spp $\sigma = 0.01$

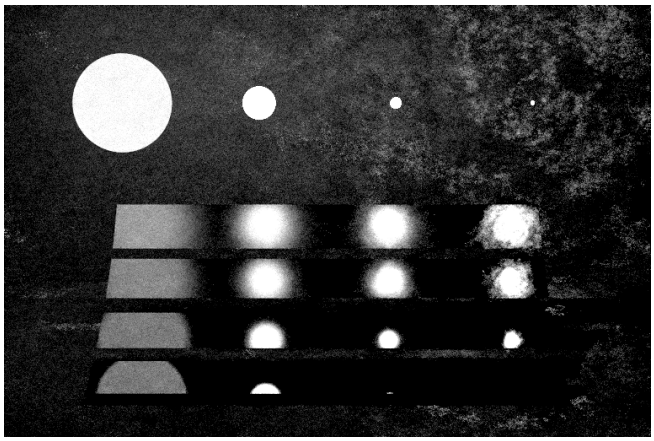

Fig. 15. Result from DRPSSMLT + Path Tracer 32spp

Tuning the sigma parameter in PSSMLT can change the performance of PSSMLT in concentrating on light region. Here are two picture showing even smaller σ values

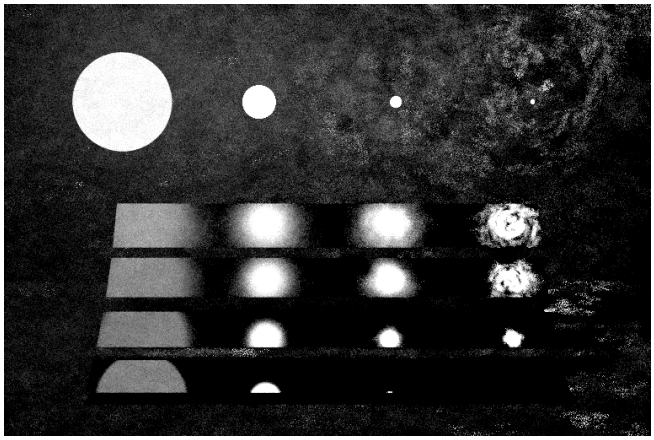

Fig. 16. Result from PSSMLT + Path Tracer 32spp $\sigma = 0.005$

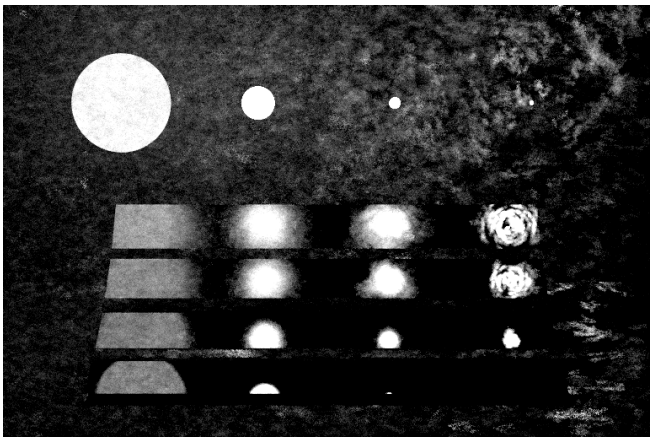

Fig. 17. Result from PSSMLT + Path Tracer 32spp $\sigma = 0.002$

We can see some improvement in the upper right light area but as a tradeoff, the background goes very noisy. A smaller value smaller than this won't make much difference.

Also, here are three path tracer pictures with higher SPP, in contrast with 2,3,4,7,8,9

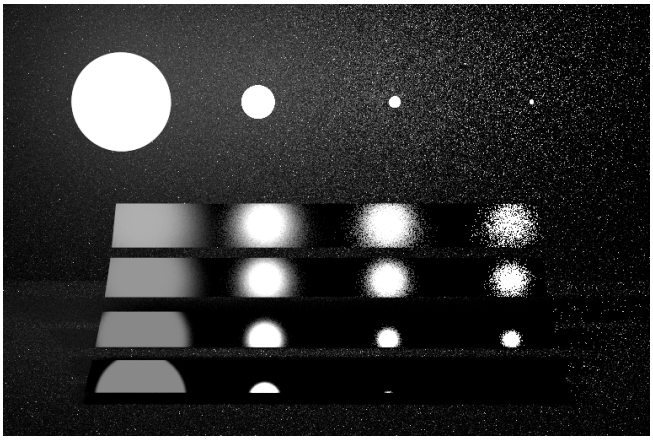

Fig. 18. Result from Path Tracer 256spp

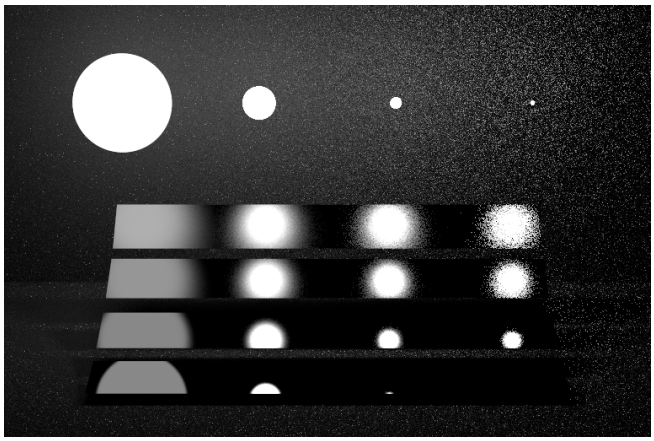

Fig. 19. Result from Path Tracer 1024spp

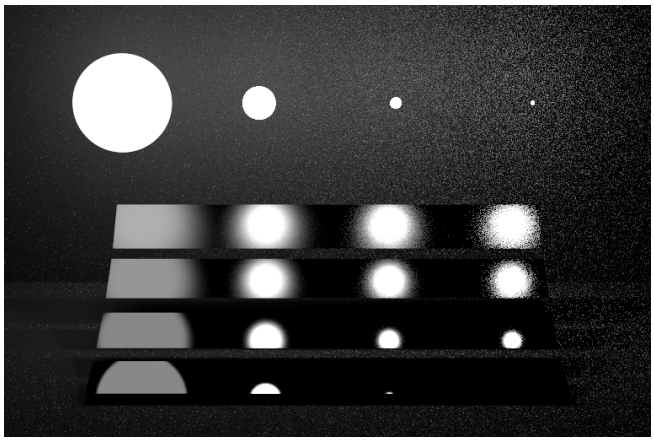

Fig. 20. Result from Path Tracer 2048spp

In conclusion, both PSSMLT and DRPSSMLT excels in focusing on light part of the picture. DRPSSMLT's second stage performs really well(much better than PSSMLT even with a very small sigma) in upper right light part of the picture. Due to the limitation of spp, the dark part of PSSMLT and DRPSSMLT will be noisier.

5.3 Motion Blur

My final image is a linearly moving toy car:

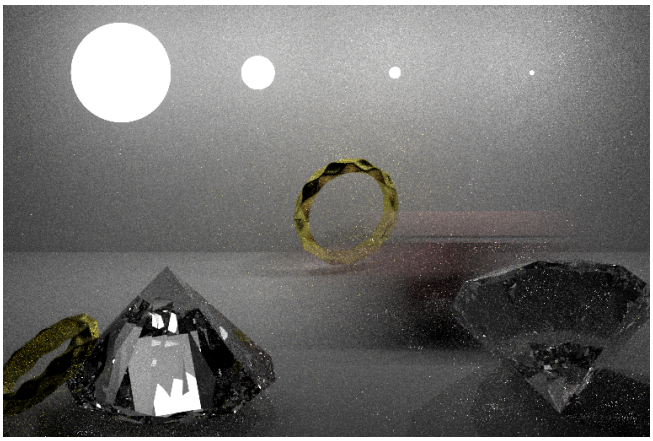

Fig. 21. A moving toy car

I also have a rotation motion blur on a checker ball, which seems really strange. I think there might be some problem that I don't quite understand.

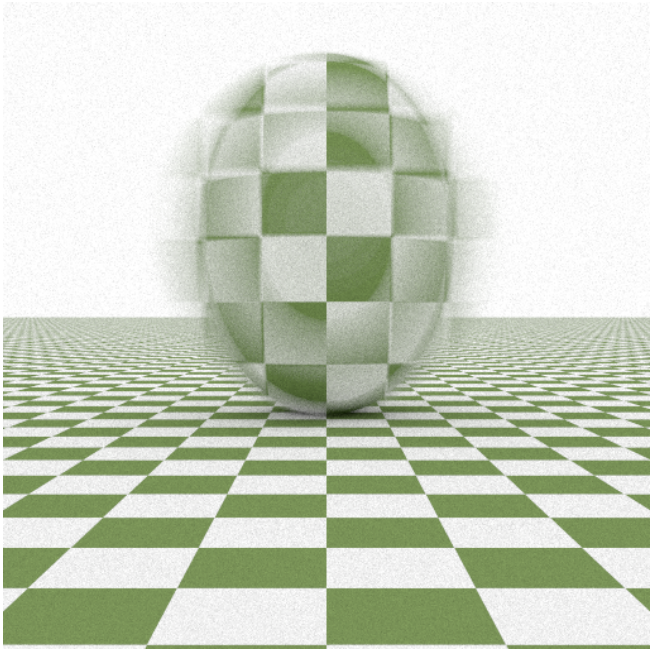

Fig. 22. A rotating ball

6 PROBLEMS AND POSSIBLE IMPROVEMENTS

6.1 Floating Point Problem in DRPSSMLT

When applying DRPSSMLT to MIS path tracer, I noticed that there could be many NaN, or near 0 results being generated. Special checks in the code are needed or it will generate black pixels to the final image. This is also observed in standalone MIS integrator. I would consider this is some unknown bugs in DIRT's reference code.

6.2 Lazy Evaluation for DRPSSMLT

The DRPSSMLT implementation right now use a fixed dimension path space. And every mutation will mutate the whole path space. This is a bad design in real application. Complicated scene need more random numbers while simple scenes need less. Lazy evaluation in DRPSSMLT is important. I did not implement this because DRPSSMLT mutate two random numbers at a time. And you also need to record the order of all first stage and second stage mutations to do the make up for the lazy evaluation.

6.3 Better Transformation Interpolation

As I mentioned, a better interpolation technique, such as Quarternion could be used to interpolate rotation. Also, the transformation right now only supports linear motion. Techniques like interpolating translation using Bezier curve are needed for non-linear motion.

6.4 Time Consumption

PSSMLT and DRPSSMLT generally take longer to run. My naive implementation of DRPSSMLT takes longest. Even with lazy evaluation. DRPSSMLT will take the longest as there is a second stage mutation. This extra time is not worth it in simple scene like the Veach scene where general unidirectional MIS is capable of making a good picture. I will wonder how PSSMLT really works on top of BDPT to handle those really tricky scenes.

6.5 Sampling time

Right now I have no way to control the luminance of the whole picture as a shutter does for a real camera. So in fact, my feature is not sampling time, but sampling the intermediate position of an object.

REFERENCES

[1] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum* 21, 3 (2002), 531–540. https://doi.org/10.1111/1467-8659.t01-1-00703 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.t01-1-00703

[2] Damien Rioux-Lavoie, Joey Litalien, Adrien Gruson, Toshiya Hachisuka, and Derek Nowrouzezahrai. 2020. Delayed Rejection Metropolis Light Transport. *ACM Trans. Graph.* 39, 3, Article 26 (may 2020), 14 pages. https://doi.org/10.1145/3388538

[3] Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 65–76. https://doi.org/10.1145/258734.258775