UCSD CSE 272 Assignment 3:

Final Project

The final project will be a project of your choice. You can implement some non-trivial rendering algorithms, or render some interesting scenes in lajolla or your own renderer. It can also be a rendering-related research project with a new algorithm or system. Below we provide some options for you to think about, but feel free to come up with your own ones.

Grading. The project will be graded based on two aspects: technical sophistication and artistic sophistication. You can have a project that does not have much technical component, but it renders beautiful images. Alternatively, you can have a project that produces programmer arts, but is highly technical. To encourage people to pursue ambitious projects, the more ambitious the project is, the less requirement that is to the completness of the project. For example, if you aim to work on a research idea that is novel enough to be published at SIGGRAPH, we will not judge you using the standard of a technical paper. It is sufficient to show your work and the initial prototypes and experiments you develop, even if the initial result does not show practical benefit yet.

Logistics. Before 2/27, please let us know what you plan to do for the final project by submitting to Canvas, and have a brief plan in one or two paragraphs. Schedule a chat with us if you have any question. We will have a checkpoint at 3/13: send us a brief progress report in a few paragraphs on Canvas (ideally provide results and images) describing what you did and what you plan to do next. Let's have a maximum of two people in a group. For the research type projects, to avoid conflicts, if multiple groups want to work on the same project, we will consider merging the groups.

1 Implementation ideas

Following are projects that are less novel, but could produce cool images. They are usually about the implementation of the papers we talked about in the class. I did not sort them in terms of difficulty.

Energy-preserving microfacet BSDFs. Implement Heitz et al.'s multiple-scattering microfacet BSDF [14] in lajolla. Alternatively, implement the position-free variant [32]. As a bonus, compare them to the energy compensation technique introduced by Kulla and Conty[1], and show the pros and cons.

Normal map and displacement map filtering. Implement LEAN [28] and use it for rendering high resolution normal map in lajolla. Compare it to a high sample count reference. When does it work well and when does it fail? As a bonus, implement LEADR [7] for filtering displacement map. First you will need to implement displacement mapping in lajolla. This can be done in a preprocessing pass to convert meshes with displacement map textures into tesselated mesh, or in an on-the-fly manner [31]. Alternatively, implement Yan et al.'s normal map filtering technique [35] in lajolla.

Layered BSDF. Implement the position-free layered BSDF simulator from Guo et al. [11] in lajolla. As a starting point, maybe start with a two layer BSDF with no volumetric scattering in between. Implementing the unidirectional version is good enough. As a bonus, read Gamboa et al.'s work [8] and implement their sampling strategy.

Hair BCSDF. Implement Marschner's hair scattering model [23] in lajolla. You'll have to implement a ray-curve intersection routine (it is acceptable to reuse the one from Embree) and a hair BCSDF. You may want to read Matt Pharr's note on implement hair rendering in pbrt.[2].

[1] https://blog.selfshadow.com/publications/s2017-shading-course/imageworks/s2017_pbs_imageworks_slides_v2.pdf

[2] See https://www.pbrt.org/hair.pdf

Thin-film iridescence BSDF. Implement Belcour's thin-film iridescence BSDF [3] in lajolla. Feel free to look at Belcour's source code.

Efficient transmittance estimation. Implement Kettunen et al.'s unbiased ray marching estimator [16]. You can extend your volumetric path tracer in homework 2 for this. Replace your ratio tracker with the new ray marching estimator. Compare to the ratio tracker and analyze their variance reduction properties.

BSSRDF. Implement a BSSRDF in lajolla. You can use the one from Christensen and Burley [6]. For importance sampling the BSSRDF, check out King et al.'s talk[3]. You are encouraged to read pbrt's code[4].

Single scattering. Implement Chen et al.'s 1D Min-Max mipmap shadow mapping technique for rendering single scattering volumes [5]. You don't have to implement it on GPUs or show real-time performance.

Stratification. Implement low-discrepancy sampling in lajolla. A simpler starting point is Halton sequence. Read the related chapters in pbrt[5] for the reference. Smallppm[6] from Toshiya Hachisuka also contains a low discrepenacy photon mapper using Halton sequence.

Bidirectional path tracing. Implement a bidirectional path tracer in lajolla. Read the related chapters in pbrt[7] for the reference. Another good reference code is *smallpssmlt*[8] from Toshiya Hachisuka.

Progressive photon mapping. Implement progressive photon mapping [12] in lajolla. Read the related chapters in pbrt[9]. Another good reference code is *smallppm*[10] from Toshiya Hachisuka. You can also implement the probabilistic variant from Knaus et al. [18].

Gradient-domain path tracing. Implement gradient-domain path tracing [17] in lajolla. A good reference code is smallgdpt from me[11].

Metropolis light transport. Implement Kelemen-style Metropolis light transport [15] in lajolla. Read the relevant chapters in pbrt[12]. Another good reference code is *smallpssmlt*[13] and *smallmmlt*[14] from Toshiya Hachisuka.

Optimal multiple importance sampling. Implement *optimal* multiple importance sampling using control variates from Kondapaneni et al. [19] in lajolla. Ideally, apply their method for a unidirectional path tracer or even a bidirectional path tracer (instead of just applying it for direct lighting) – what kind of data structure do you need for maintaining the required statistics?

Lightcuts. Implement stochastic lightcuts [36] from Yuksel et al. in lajolla for rendering scenes with millions of lights. As a bonus, implement the data-driven version from Wang et al. [33].

ReSTIR. Implement ReSTIR [4] in lajolla for rendering scenes with many lights. You don't need to implement the temporal reuse, but you should explore spatial reuse.

[3] https://pdfs.semanticscholar.org/90da/5211ce2a6f63d50b8616736c393aaf8bf4ca.pdf
[4] https://www.pbr-book.org/3ed-2018/Light_Transport_II_Volume_Rendering/Sampling_Subsurface_Reflection_Functions
[5] https://www.pbr-book.org/3ed-2018/Sampling_and_Reconstruction/The_Halton_Sampler
[6] https://cs.uwaterloo.ca/~thachisu/smallppm_exp.cpp
[7] https://www.pbr-book.org/3ed-2018/Light_Transport_III_Bidirectional_Methods/Bidirectional_Path_Tracing
[8] https://cs.uwaterloo.ca/~thachisu/smallpssmlt.cpp
[9] https://www.pbr-book.org/3ed-2018/Light_Transport_III_Bidirectional_Methods/Stochastic_Progressive_Photon_Mapping
[10] https://cs.uwaterloo.ca/~thachisu/smallppm_exp.cpp
[11] https://gist.github.com/BachiLi/4f5c6e5a4fef5773dab1
[12] https://www.pbr-book.org/3ed-2018/Light_Transport_III_Bidirectional_Methods/Metropolis_Light_Transport
[13] https://cs.uwaterloo.ca/~thachisu/smallpssmlt.cpp
[14] https://cs.uwaterloo.ca/~thachisu/smallmmlt.cpp

GPU rendering. Port most of lajolla to Vulkan/DirectX 12/CUDA/OpenCL/Metal.

2 Research project ideas

These projects are likely publishable in a conference or a journal if done well (you will likely need to work on it longer even after the course ends for this to happen though). They are for students who are more motivated and want to get into rendering research. If you choose to work on these projects, it is possible that they will not be finished at the end of the quarter. This is totally fine and you will still get high/full points if you show your work. We are happy to work with you after the quarter to finish the project if you did well and are interested.

For anything below, feel free to reach out to us for clarification about details of the project. I did not sort them in terms of difficulty.

Differentiable hair rendering. Existing inverse hair rendering methods (e.g., [29]) do not consider the light scattering in the hair fiber, and this can lead to less realistic results. In this project, we will develop a differentiable renderer [22] that is capable of computing the derivatives of a rendering of hair with respect to the shading and geometry parameters, and use it for inferring hair reflectance and geometry from images. I would recommend porting pbrt's hair rendering code[15] to Mitsuba 3[16]'s automatic differentiation utility for derivative computation. For the scope of the final project, it is sufficient to only consider the hair BSDF parameters and ignore the curve geometry. To handle the geometry derivatives, you can likely adapt Bangaru et al.'s differentiable signed distance field rendering algorithm [2] (since the computation of ray-curve intersection relies on the distance between the ray and the curve).

Guided tri-directional path tracing. From Veach's path-space formulation, it is clear that in addition to tracing rays starting from eyes or lights, it should also be possible to trace rays starting from an arbitrary point in the scene (we explored a similar method in 2017 [1]). However, such *tri*-directional path tracing method is difficult to apply in practice since it is difficult to decide where should we start tracing the ray. In this project, we will design a path guiding algorithm [20, 25] that builds a 5D data structure to decide which point and direction the path should start with. In the first phase, we will start tracing rays randomly, and record their contributions in the 5D data structure. We will then importance sample the recorded contribution and iteratively update the data structure. Similar methods have been applied to differentiable rendering (e.g., [34]), but it has not been applied in forward rendering algorithms yet. For the scope of the final project, it is not necessary to combine your method with the camera and light subpaths. Ultimately, it would be super cool to combine the tri-directional path tracer with VCM/UPS and use data to decide with subpath to use [9].

Motion-aware path guiding for animation rendering. Path guiding algorithms build data structures that record contributions of light paths and use them for importance sampling. These data structures typically are fitted in a per-frame basis and do not easily extend to animation rendering. In this project, we will explore ways to update the path guiding data structures over time. A potential point of reference is the neural radiance caching work from Muller et al. [27]. A relatively simple starting point is to implement a simple exponential weighted moving average update for the 5D tree data structure of Muller et al. [25]. Can you come up with heuristics to account for occlusion and parallax? What kind of information we can extract from a renderer that can help updating the data structure?

Denoising Metropolis light transport rendering. Denoising Monte Carlo rendered images have been extensively used in production rendering. However, denoising of Metropolis light transport rendered images have not been looked at at all. In particular, Metropolis light transport methods tend to produce correlation artifacts in images (e.g., patches of images being much brighter than it should), that can be jarring especially for animation rendering. Can we design denoising algorithms that work well for Metropolis light transport

[15] https://www.pbrt.org/hair.pdf
[16] https://github.com/mitsuba-renderer/mitsuba3

rendering? A relatively simple starting point is to collect a bunch of images rendered using Metropolis light transport and use them to train a standard convolutional neural network. How do we ensure the output animation to be temporally stable? Can we output anything during rendering to help denoising? A relevant reference is the work from Gu et al. [10] for combining a path traced image and a denoised image with correlated artifacts. We can use a similar technique to combine a path traced image (can be collected "for-free" during the large step mutation in a Kelemen-style renderer [15]) and the correlated MLT rendering.

Neural mutation for Metropolis light transport. Neural networks have been shown to be helpful for importance sampling in a Monte Carlo path tracer [26]. Can we apply them to Metropolis light transport as well? Similar methods have been explored in the machine learning community for Hamiltonian Monte Carlo [21]. In this project, we will investigate the use of neural networks for designing mutation in a Metropolis light transport renderer. A relatively simple starting point is to implement Levy et al.'s method in a Kelemen-style path tracer and see how well it works.

Multiple-scattering NeRF with neural scattering fields. Neural Radiance Fields [24] and their variants have been revolutionizing 3D computer vision using inverse volume rendering. However, all of the current NeRF variants do not consider multiple-scattering within the volumes. In this project, we will explore the incorporation of multiple scattering in neural fields. To achieve this, we need a neural representation that can represent a spatially varying, anisotropoic scattering coefficients and phase function. We will also need a way to importance sample both the phase function and transmittance of this neural representation. Alternatively, we can also explore a voxel-based representation (which can have similar performance to a neural representation! [30]) using the SGGX phase function [13] and spherical harmonics for the scattering coefficients. I would recommend starting from known lighting (e.g., an environment map) and single scattering to make the problem easier.

Spatially varying neural BSDF for neural signed distance fields. Another popular neural scene representation is neural signed distance fields [2]. However, similar to the situation of NeRF, existing neural signed distance fields do not model the BSDF of the surfaces (they assuming all surfaces are pure emitters). In this project, we will explore modeling a spatially varying BSDF for neural signed distance fields. The network architecture can be a 7D coordinate network that takes a 3D position and two 2D directions and outputs the BSDF response. For importance sampling, it might be useful to make the network a normalizing flow [26] conditioned on the position and incoming direction. Similar to above, I would recommend starting from known lighting (e.g., an environment map) and direct lighting to make the problem easier.

References

[1] Luke Anderson, Tzu-Mao Li, Jaakko Lehtinen, and Frédo Durand. Aether: An embedded domain specific sampling language for Monte Carlo rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 36(4):99:1–99:16, 2017.

[2] Sai Bangaru, Michael Gharbi, Tzu-Mao Li, Fujun Luan, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. Differentiable rendering of neural SDFs through reparameterization. In *ACM SIGGRAPH Asia Conference Proceedings*, 2022.

[3] Laurent Belcour and Pascal Barla. A practical extension to microfacet theory for the modeling of varying iridescence. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 36(4):1–14, 2017.

[4] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 39(4):148, 2020.

[5] Jiawen Chen, Ilya Baran, Frédo Durand, and Wojciech Jarosz. Real-time volumetric shadows using 1D min-max mipmaps. In *Symposium on Interactive 3D Graphics and Games*, pages 39–46, 2011.

[6] Per H. Christensen. An approximate reflectance profile for efficient subsurface scattering. In *ACM SIGGRAPH Talks*, 2015.

[7] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 32(6):1–11, 2013.

[8] Luis E Gamboa, Adrien Gruson, and Derek Nowrouzezahrai. An efficient transport estimator for complex layered materials. *Comput. Graph. Forum (Proc. Eurographics)*, 39(2):363–371, 2020.

[9] Pascal Grittmann, Ömercan Yazici, Iliyan Georgiev, and Philipp Slusallek. Efficiency-aware multiple importance sampling for bidirectional rendering algorithms. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 41(4), 2022.

[10] Jeongmin Gu, Jose A Iglesias-Guitian, and Bochang Moon. Neural James-Stein combiner for unbiased and biased renderings. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 41(6), 2022.

[11] Yu Guo, Miloš Hašan, and Shuang Zhao. Position-free Monte Carlo simulation for arbitrary layered bsdfs. *ACM Transactions on Graphics (ToG)*, 37(6):1–14, 2018.

[12] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 27(5):130:1–130:8, 2008.

[13] Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. The sggx microflake distribution. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 34(4):1–11, 2015.

[14] Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. Multiple-scattering microfacet BSDFs with the Smith model. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 35(4):1–14, 2016.

[15] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Comput. Graph. Forum (Proc. Eurographics)*, 21(3):531–540, 2002.

[16] Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. An unbiased ray-marching transmittance estimator. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 40(4), 2021.

[17] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain path tracing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 34(4):123:1–123:13, 2015.

[18] Claude Knaus and Matthias Zwicker. Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph.*, 30(3):25:1–25:13, 2011.

[19] Ivo Kondapaneni, Petr Vévoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. Optimal multiple importance sampling. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 38(4):1–14, 2019.

[20] Eric P. Lafortune and Yves D. Willems. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Rendering Techniques (Proc. EGWR)*, pages 11–20, 1995.

[21] Daniel Levy, Matt D. Hoffman, and Jascha Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with neural networks. In *International Conference on Learning Representations*, 2018.

[22] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[23] Stephen R Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. Light scattering from human hair fibers. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22(3):780–791, 2003.

[24] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[25] Thomas Müller, Markus Gross, and Jan Novák. Practical path guiding for efficient light-transport simulation. *Comput. Graph. Forum (Proc. EGSR)*, 36(4):91–100, 2017.

[26] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Trans. Graph.*, 38(5):1–19, 2019.

[27] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 40(4):36:1–36:16, 2021.

[28] Marc Olano and Dan Baker. Lean mapping. In *Symposium on Interactive 3D Graphics and Games*, pages 181–188, 2010.

[29] Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. Neural strands: Learning hair geometry and appearance from multi-view images. In *European Conference on Computer Vision*, pages 73–89, 2022.

[30] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Computer Vision and Pattern Recognition*, 2022.

[31] Theo Thonat, Francois Beaune, Xin Sun, Nathan Carr, and Tamy Boubekeur. Tessellation-free displacement mapping for ray tracing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 40(6):1–16, 2021.

[32] Beibei Wang, Wenhua Jin, Jiahui Fan, Jian Yang, Nicolas Holzschuch, and Ling-Qi Yan. Position-free multiple-bounce computations for Smith microfacet BSDFs. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 41(4):134:1–134:14, 2022.

[33] Yu-Chen Wang, Yu-Ting Wu, Tzu-Mao Li, and Yung-Yu Chuang. Learning to cluster for rendering with many lights. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 40(6):1–10, 2021.

[34] Kai Yan, Christoph Lassner, Brian Budge, Zhao Dong, and Shuang Zhao. Efficient estimation of boundary integrals for path-space differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 41(4):123:1–123:13, 2022.

[35] Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33(4), 2014.

[36] Cem Yuksel. Stochastic lightcuts. In *High-Performance Graphics*, pages 27–32, 2019.