Research Outline

Shape From Tracing: Reconstructing 3D Geometry and SVBRDF Material from Images via Differentiable Pathtracing

Purvi Goel

Overview

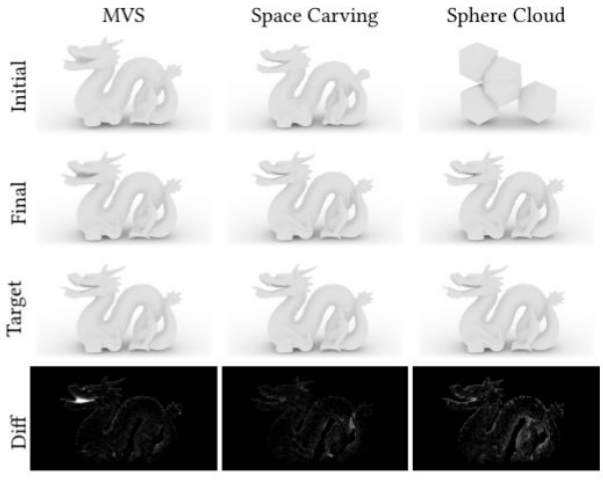
The goal of 3D reconstruction is to generate a plausible and accurate 3D model from a set of photographs. Classic approaches like voxel carving cannot accurately estimate texture, and often rely on the assumption that the object in the scene is diffuse. Optimization strategies like differentiable rasterization cannot represent complex global illumination effects like color bleeding. We pose this problem in the context of differentiable pathtracing. Using a series of input pictures of an object, we determine the object's shape and material properties through an optimization process driven by gradient descent. Differentiable rendering can reconstruct objects and scenes with difficult reflectance properties like spatially varying BRDFs, as well as complex geometries. We use a differentiable pathtracer called *Redner* [1].

While differentiable pathtracing can give us gradients with respect to a loss between current renders and target images, vanilla gradient-based optimization is extremely prone to local minima. In the context of inverse rendering, this can manifest as "messy" geometry: intersecting edges, triangles jutting into the mesh, and so on. Part of the challenge in designing the pipeline was figuring out a way to either skirt around these local minima, gradually increase the size of the parameter space in a coarse-to-fine manner, or pause the optimization to rescue the mesh from local minima.

Geometry Initializations

In order to stress-test this geometry-reconstruction strategy, I experimented with different initial geometry guesses. The most reliable way to estimate correct geometry is to start with an initial guess that is very close in shape and genus to the goal mesh. However, because our pipeline is designed to recover from local minima, we can relax this constraint and get reasonable reconstructions from the following initalizations:

- Space carving [2], a form of screen-to-world reprojection of object masks that provides a decent reconstruction if there are accurately calibrated and sufficient cameras.
- Multiview Stereo reconstruction with COLMAP [3,4], which does not require calibrated cameras but often difficult to use on simulated data.
- Sphere clouds, in which we optimize the position and scales of several small spheres to lay down very approximate geometry



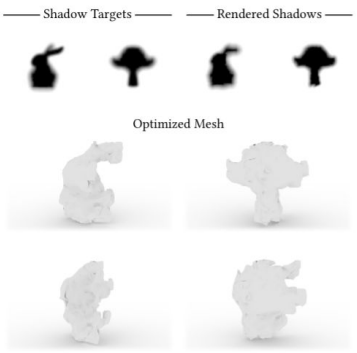
density shere the target model is located. This is the sparsest of initializations, and requires the most time in the optimization process to reconstruct target geometry. As a result, I decided that the previous two approaches were better-suited for our task. However, certain real-world target geometries are impossible to capture with MVS, such as transparent ones. Sphere clouds would be a reasonable initialization for future extensions of the project, for reconstructing highly specular or glass materials .

Texture Initialization

In early versions of the pipeline, I found that texture optimization would too-often “hide” incorrect geometry. This could happen both ways as well; geometry optimization could try to mimic details in an object’s texture. We could generally get away with the latter; smoothing steps in the geometry reconstruction pipeline such as Poisson surface reconstruction and adaptive remeshing generally would remove these inaccuracies, but we didn’t have a similar fallback for texture optimization. At first, I believed this was the result of not having enough views of the target object, or a case of sub-optimal hyperparameters. However, adding more camera angles and light positions failed to fix the problem, and a lengthy search over hyperparameters was similarly unsuccessful. I found that our texture initialization, which defined a color per vertex, provided too large a parameter space for early stages in the pipeline. When I changed the texture initialization to be much more coarse: optimizing a single diffuse, specular, and roughness value for the entire mesh for the first two rounds of optimization, the texture-baking problem disappeared.

Benefits of Indirect Illumination

Unlike differentiable rasterization, differentiable pathtracing can take advantage of indirect illumination effects within a scene. To highlight this, I created an experiment to reconstruct an object entirely by comparing shadows that it casts onto two perpendicular walls to two target shadows. Due to the regular surface resampling steps, the mesh maintains a relatively smooth surface despite the sparse target information. Note that because our pipeline produces physically-based shadows, our targets can contain lighter soft shadows. This is not possible using previous space-carving based approaches [5].



Real-world Data

This semester I worked on applying this pipeline to real-world data. Pathtracing is a physically-based illumination model. My hypothesis was that this would make our pipeline well-suited for estimating geometry and appearance from real-world photographs. We reconstruct object geometry and estimate camera poses from a sequence of video frames using COLMAP. We use an environment map reconstruction to model scene lighting.

This is a challenging problem because real-world data is bound to introduce some error, and optimizing for object properties like specular maps with differentiable pathtracing is quite dependent on accurate scene lighting. In order to build some tolerance to an imperfectly oriented environment map or minor differences in light intensities, we start with a low-resolution version of the environment map and gradually increase its resolution through the optimization process.

While we were able to reconstruct fine-scaled geometry detail in simulation using varied positions of area lights, this scene is restricted to a single lighting condition. As a result, shading and specular highlights have a tendency to get baked into diffuse albedo. To account for this, I found that it was more necessary to introduce some priors onto the scene about the variance across different materials in an object.

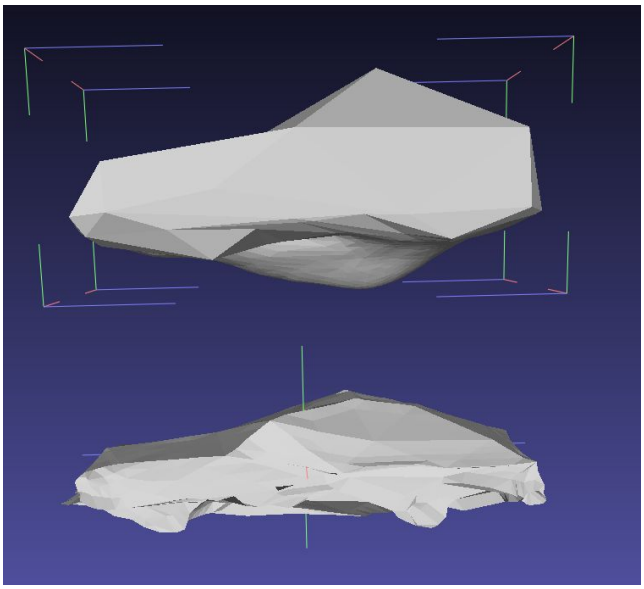
A valuable experiment that is optimizing for every material in a scene separately, following the coarse-to-fine guidelines of our initial pipeline. This is reminiscent of earlier efforts in determining the correct texture initialization in simulation. I found that we were more successful when we first optimized for single diffuse, specular, and roughness texture values for the entire mesh before expanding the texture parameter space. However, our objects in simulation do not exhibit a lot of variance in their spatially-varying albedos. One of our real-world reconstruction cases, shown on the right, contained two noticeably different materials. I decided to segment the materials early in the pipeline. I optimized a single color for every vertex, then ran a clustering algorithm on these values. Every cluster contained vertices that exhibited similar BRDFs. By reassigning the vertex colors to only take on values belonging to this “palette” in early stages of initialization, I could still maintain a coarse texture estimation while accounting for different materials in the scene.

Per-material texture optimization has greatly reduced the amount of reflectance-baking that the texture maps exhibited. There is a room for future exploration into choosing the number of clusters and reliable, automatic segmentation of materials.

Finally, the initial guess of the material maps has an effect on how much shading and specular reflections are baked into diffuse maps. I found that, when given insufficient views and lighting conditions, the optimizer is much better at “hiding” detail within diffuse texture rather than explaining detail with specular texture. This even occurs when optimizing texture per material. The solution I found was to introduce a bias towards specular texture by initializing all diffuse colors to black, and all specular colors to white (this turns the object into a near-perfect mirror). This way, reflections of the environment already exist in the renders, and can’t be baked into diffuse texture maps.

Differentiable Pathtracing as Refinement for Reconstruction with Deep Learning

Recent advances in 3D deep learning have also been made to reconstruct objects from photographs. The results remain somewhat unsatisfying; generally these methods cannot construct complex appearance models, and fail to capture fine geometry details. They also tend to overfit to training data, and have trouble generalizing to test data with different objects, or similar objects with different poses, than the training set.



However, neural network reconstructions can be used as an initialization for our pipeline. I used Neural Mesh Renderer [6], a single-view reconstruction network to reconstruct the car on the right (top) from an image (left). Feeding this into our pipeline, I successfully refined the very coarse initialization (bottom). Details emerge in the wheels, hood of the car, and details in the front and back bumpers.

The next step is using a multiview-reconstruction deep network. I used Pixel2Mesh++ [7] to reconstruct an object given 3 target views. Similarly to Neural Mesh Renderer, Pixel2Mesh++ cannot estimate colors, and output a diffuse gray approximation of the texture and very coarse geometry reconstruction. I found that our pipeline can handle this initialization as well to recover a finer shape and texture estimation.

My opinion is that differentiable pathtracing is currently too slow to be added to a deep network's training loop. However, this experiment suggests that after inevitable future advances in speeding up and lowering the variance in a differentiable pathtracer's gradients, it can be an elegant addition to a neural architecture.

Limitations

One of the main limitations of this pipeline is its dependence on accurate scene conditions. A simulated light source within a rendered scene undergoing optimization that is less emissive than it is within a target scene will inevitably result in a somewhat inaccurate reconstruction. While physically-based lighting models are powerful for modeling intricate interactions between light and shapes within scenes, even small changes in the scene set-up can cause significant changes in renders. This limitation became quite relevant when using real-world data, as there was no "ground-truth" simulated scene.

Another limitation is the amount of time that optimization takes. Pathtracing is a greedy, time-intensive process and rendering between sixteen and sixty-four images per iteration can take a significant amount of time.

Future Work

Interesting future directions would be using differentiable pathtracing to reconstruct objects with more complex materials, such as glass, and those exhibiting subsurface scattering or anisotropic reflections. It would also be worthwhile to explore different strategies for choosing which camera frames are most helpful for a given epoch.

Image Acknowledgements: Figures in this report are from a paper written jointly with fellow students Loudon Cohen and Brad Guesman, and Professors Daniel Ritchie and James Tompkin.

[1] Li, Tzu-Mao et al. “Differentiable Monte Carlo ray tracing through edge sampling.” *ACM Transactions on Graphics (TOG)* 37 (2018): 1 - 11.
[2] Kutulakos, Kiriakos N. and Steven M. Seitz. “A Theory of Shape by Space Carving.” *International Journal of Computer Vision* 38 (2000): 199-218.
[3] Schönberger, Johannes L. and Jan-Michael Frahm. “Structure-from-Motion Revisited.” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016): 4104-4113.
[4] Schönberger, Johannes L. et al. “Pixelwise View Selection for Unstructured Multi-View Stereo.” *ECCV* (2016).
[5] Mitra, Niloy Jyoti and Mark Pauly. “Shadow art.” *SIGGRAPH 2009* (2009).
[6] Kato, Hiroharu et al. “Neural 3D Mesh Renderer.” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018): 3907-3916.
[7] Wen, Chao et al. “Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation.” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019): 1042-1051.