CSE 274: Introduction

Lecturer: Ravi Ramamoorthi

Scribed by Owen Jow on September 27, 2018

1 Motivation: High-Dimensional Appearance

Photorealistic rendering often requires us to collect data which varies across many dimensions.

- For a variant of *precomputed real-time rendering*,[1] we want to store the light transport matrix $\mathbf{T}$ which encodes reflection, scattering, etc. in the scene (as described by, e.g., the rendering equation or the BSSRDF or whatever one wants to support) over lighting directions and viewing directions and/or surface positions, and can be multiplied by a lighting distribution (e.g. incident illumination) L as $B = \mathbf{T}L$ to compute the final image/radiance values B.

  With lighting directions, viewing directions, and surface positions, this is 4-6 dimensions.

- For *Monte Carlo rendering (path tracing)*, we want to sample *at each pixel* illumination values corresponding to different times, lightings, and lens apertures (the latter for depth of field). This is anywhere from 3 to 7 dimensions of variation.
- For *appearance acquisition* (acquiring reflectance functions, e.g. using a spherical gantry), we have anywhere from 4 to 8 dimensions, with e.g. a SVBRDF or a BSSRDF. Usually we will want to sample over at least lighting direction and viewpoint.
- For *computer vision*, when accounting for different lighting conditions we can easily end up with 4 dimensions: two for the lighting direction, two for the image position.
- For *motion blur rendering*, we need to sample over both spatial extent *and* temporal variation at each pixel. (Without motion blur, we get an unpleasant strobing effect.) Three dimensions!
- For *real-time global illumination*, we have to consider two dimensions of incoming directions across two dimensions on the scene. And two plus two equals...
- Appearance has increasingly been represented as volume(s), instead of simply a surface BRDF. This yields a "fluffy appearance" but requires e.g. a $1000 \times 1000 \times 1000$ voxel grid instead of a 1000×1000 image.
- For *lighting-insensitive recognition*, we can illuminate a subject within a light stage from many different incident directions in order to register its (his/her) appearance better. The lighting here can be represented as a 5D subspace.

Even if we only take 100 samples per dimension, we quickly run up against the curse of dimensionality. With six dimensions, 100 samples per dimension is 100^6 samples in total!

[1] *Precomputed real-time rendering*: store light transport data so as to interactively change lighting/viewpoint.

To get around this, we can sample sparsely and *reconstruct* the image we would have gotten if we had sampled abundantly. This method uses the same offline Monte Carlo rendering algorithm as always, but with "smart sampling and filtering" to avoid sampling too much.

What do we mean by "smart?" One idea is to *adaptively sample.* Take more samples in high-frequency regions and filter less; take less samples in low-frequency regions and filter more.

In summary, it's hard to solve many problems directly, because their high dimensionalities would create extreme computational costs. The problem is the sampling; we have to avoid high sampling rates even if the sampling is followed by compression. Therefore the goal is to sample sparsely and efficienctly and reconstruct a high-quality image from that sparse representation.

Applications include ***Monte Carlo rendering with sampling and denoising*** [2] *(a cornerstone in production and real-time rendering pipelines),* ***appearance acquisition and editing****, and* ***computational photography****.*[3]

2 Historical Development

2.1 Data-Driven Visual Appearance

Since ∼1993, there has been a surge of interest in data-driven visual appearance, as opposed to the previous approach of painstakingly modeling the geometry of scenes by hand (alongside simple reflectance models such as Blinn-Phong and Torrance-Sparrow). However, with data-driven methods, quality of results depends on quality of input data, in this case geometry and reflectance models.

To maximize quality, the idea was to look at data in the real world and emulate that same photo-realistic appearance during rendering. By the mid '90s, we had a version of *image-based rendering* i.e. taking a bunch of images of a scene and moving between them. Later, we would also apply data-driven approaches to problems such as synthetic relighting.[4]

We can acquire geometry using a 3D scanner or **reflectance** [5] using a gantry. We can also measure **lighting** as *environment maps* by placing a light probe in a scene and then taking a photo which captures all of the lighting in the scene ("image-based lighting").

Types of visual appearance that we can measure include reflectance fields, subsurface scattering,[6] volumetric scattering,[7] bidirectional texture functions,[8] and time-varying surface appearance.[9]

3 General Plenoptic Function

The general plenoptic function gives all knowledge of light flowing through the scene. Light can flow through any point in space (x, y, z), in any direction (θ, ϕ), at any time t, and at any wavelength λ.

[2]Take a sparse sampling step, then feed the output to a denoiser.

[3]Apply methods to images taken by a camera.

[4]Create a realistic appearance of a synthetic scene under new illumination conditions.

[5]We can model this using a BRDF, which (as part of appearance) says how a surface reflects light.

[6]Light hits a surface, scatters, and comes out at different point on the surface.

[7]e.g. clouds

[8]Appearance of texture parameterized by viewing and lighting directions.

[9]e.g. time extrapolation of a footprint drying, modeled perhaps using a Space-Time Appearance Factorization

The 7D [10] general plenoptic function is

$$f(x, y, z, \theta, \phi, t, \lambda)$$

The 14D general *scattering* function is

$$f(x_i, y_i, z_i, \theta_i, \phi_i, t_i, \lambda_i; x_o, y_o, z_o, \theta_o, \phi_o, t_o, \lambda_o)$$

and describes the relationship between incoming and outgoing light. (In other words, it takes one plenoptic function and returns another plenoptic function.) If we have one plenoptic direction, what fraction of energy goes out in this other plenoptic direction?

14 is a lot of dimensions. We can simplify by

- ignoring time dependence, thus losing time-varying BRDF properties and phosphorescence [11]
- ignoring wavelength (e.g. representing with RGB colors only), thereby losing fluorescence
- parameterizing position on a plane

i.e. dropping t, λ, and z. And so we are left with a 4D plenoptic function and an 8D scattering function. We call this 8D scattering function the BSSRDF (*bidirectional surface scattering reflectance distribution function*). It allows light to come in at one location on a surface, scatter, and leave at a different location in a different direction.

BSSRDF: "What fraction of the light that comes in here from this direction scatters this way?"

3.1 Taxonomy of Appearance

As we've seen, we can ignore certain inputs to obtain the 8D BSSRDF from the 14D scattering function. We don't need to stop there; the BSSRDF can be broken up into many other functions as well. Fundamentally, it has four inputs: $\mathbf{x}_i$, $\mathbf{x}_o$, ω_i, and ω_o. By choosing different selections of these four inputs, we can define many different appearance functions.

- By ignoring subsurface scattering (point of entry is point of exitance), we go from the BSSRDF to the 6D spatially-varying BRDF (SVBRDF) $f(x, y, \theta_i, \phi_i, \theta_o, \phi_o)$.
- By ignoring lighting direction, we go from the SVBRDF to the 4D surface light field $f(x, y, \theta, \phi)$.

We can create 15 [12] different functions by varying the four inputs of the BSSRDF. Most have some reasonable physical meaning.

Since high dimensionality is such a problem, we like to factor the high-dimensional functions into lower-dimensional functions (e.g. into *not time-varying*). For example, we can factor 6D transport $T(\mathbf{x}, \omega_i, \omega_o)$ into a 4D visibility representation [13] $V(\mathbf{x}, \omega)$ times a 4D BRDF $\rho(\omega_i, \omega_o)$. Then we only have to take, say, 10^8 samples for two 4D functions instead of 10^{12} samples for one 6D function.

Note: the previous factorization example can be described using **cubemaps**. In 6D transport, we're looking at the scene from all viewing directions at all geometric locations, and wondering how they respond to light that can come from any direction. To represent an instance of this for one viewing direction and surface position, we can consider directions on the unit sphere, projected onto the surface of a cube. The faces of the cube, unwrapped, form a *cubemap*.

[10] This is the simple version. You can add more dimensions, e.g. relating to polarization state and quantum effects.

[11] *Phosphorescence*: illuminate at one time instance, glow at a different time instance.

[12] We have a choice of using or not using each of four inputs, giving 2^4 functions (but we ignore the empty function).

[13] This is the geometry representation. Which directions are blocked? Which directions are visible?

3.2 Triple Product Integral Relighting

In triple product integral relighting, we can take the product of a *lighting cubemap* (independent of geometry and materials), a *visibility cubemap* (computed for geometry, independent of materials), and a *BRDF* [14] (computed for materials, independent of geometry). Note that the product of the latter two terms is the 6D transport, as just discussed.

Lighting environment maps are independent of where you are in the scene, since lights are assumed to be far away. They describe the variation of lighting in different directions, and only need to be computed once per frame (i.e. they are cheap).

To compute the appearance at a point, we integrate the triple product over all incident directions. To relight images, we can simply modify the lighting cubemap. Note that in this scenario, the data is still acquired via brute force before doing factorization.

4 Current Development

The main application area of appearance sampling/reconstruction is Monte Carlo GI [15] rendering. In fact, the primary development that has made path tracing practical for production rendering is sampling and reconstruction, which has provided a 1-2 order of magnitude speedup.

The latest developments use ML [16] for reconstructing sparsely sampled representations. (NVIDIA OptiX 5.0 even includes an AI-accelerated denoiser!) With ML, we can go as low as 1 spp [17] and it still works. Basically, with smart sampling and reconstruction we can do accurate physically-based rendering at insanely low sample counts.

With a combination of NVIDIA's new Turing architecture (which can trace allegedly 10 billion rays per second) and sampling/reconstruction, we can produce amazing GI demos in real time.

There is also work in fast/sparse precomputation of light transport.

[14] A function of the lighting and viewpoint directions. Precomputed for a small set of materials in the scene.

[15] Have both direct and indirect lighting. Gives inter-reflections, depth of field, soft shadows...

[16] e.g. CNNs applied to image synthesis. There is a convergence of [both real-time and offline] rendering and ML.

[17] Input to denoiser: a 1 spp rendering. Use, e.g., a learning-based filter or recurrent autoencoder to denoise.