MIP-Mapping Reflectance Models

MIP-Mapping Reflectance Models

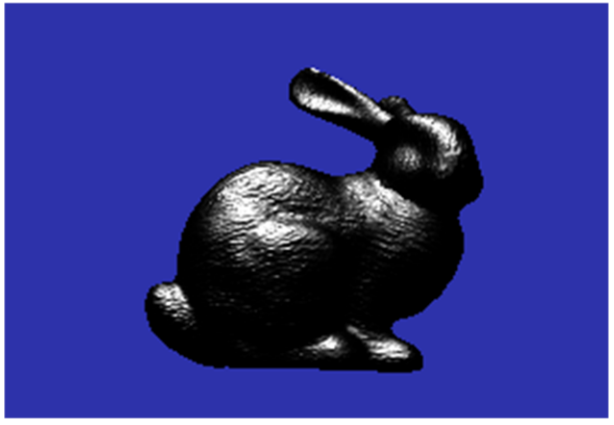
- Commonly thought of as the normal map filtering problem
- Some common techniques involve first MIP mapping the height map, and then creating a normal map from them, or fading the normal to (0,0,1)
- All of these techniques are grossly inaccurate– in fact, the normal is the least important aspect of a correct MIP filter
- Only NDM (Olano and North) filters linearly

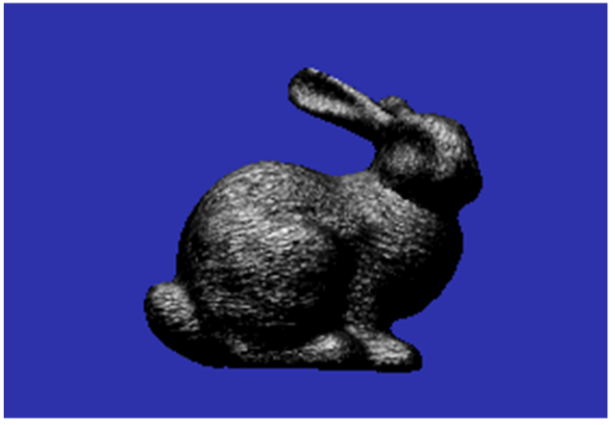
A simple non linear function

$$f(L,V) = k_d(N \cdot L) + p * k_s(N \cdot |L+V|)^p$$

The Blinn-Phong lighting model is one of the most common half angle based lighting functions. The above is a normalized version, that is the total emitted energy is close to constant for any power. This model is parameterized by N, Kd, Ks, and p. Changing these values changes the properties of the underlying surface

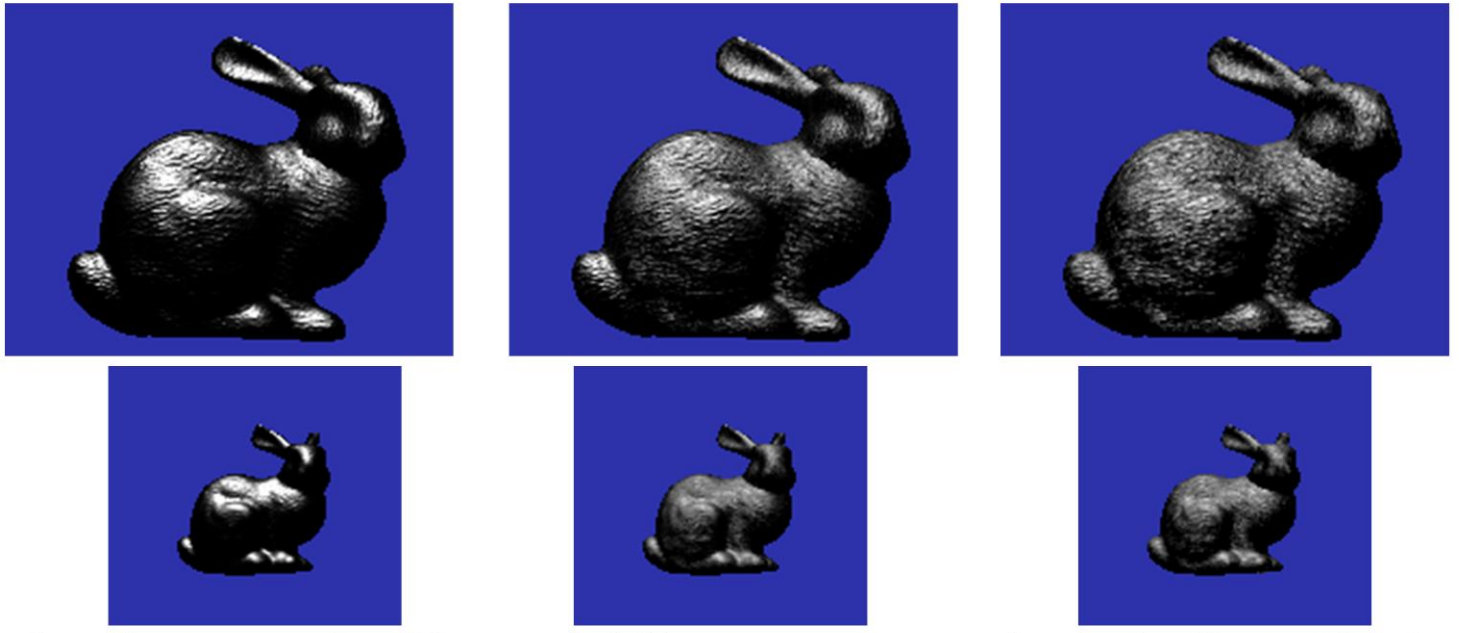
A common approach





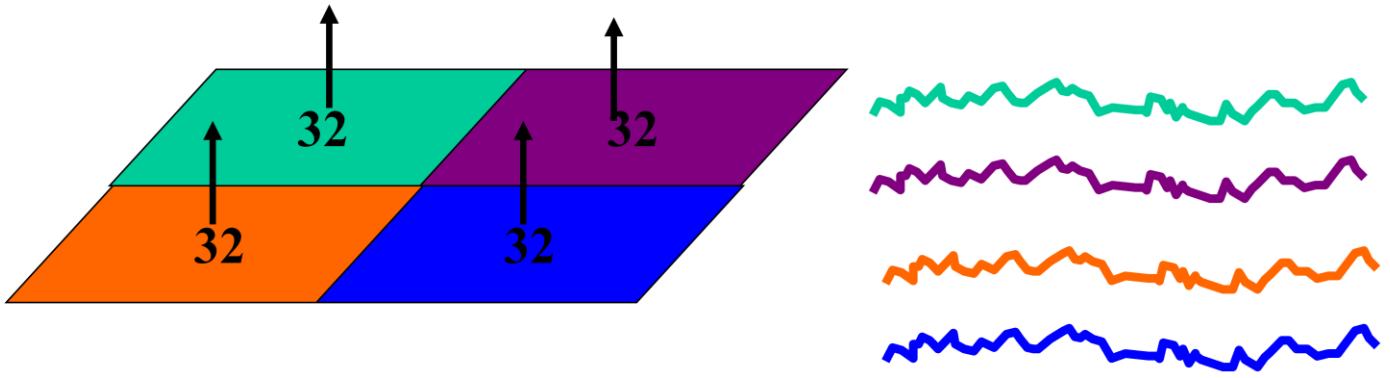
The image on the left is lit with MIP maps fading to flat, while the image on the right is what the image would look like if rendered at a higher resolution and scaled down. The objects do not physically look the same.

A more correct MIP map

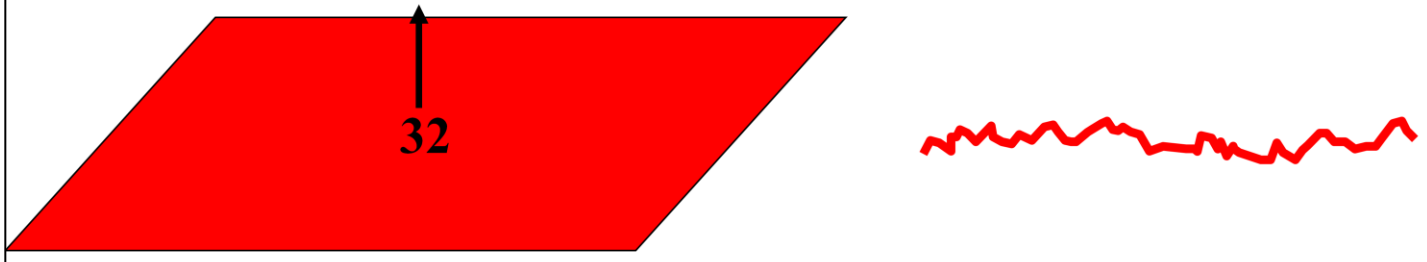


The images in the middle were rendered directly to the screen using a BRDF approximating MIP map. The more we zoom out, the larger the difference between the correct and incorrect approaches

Examining a mip level - simple

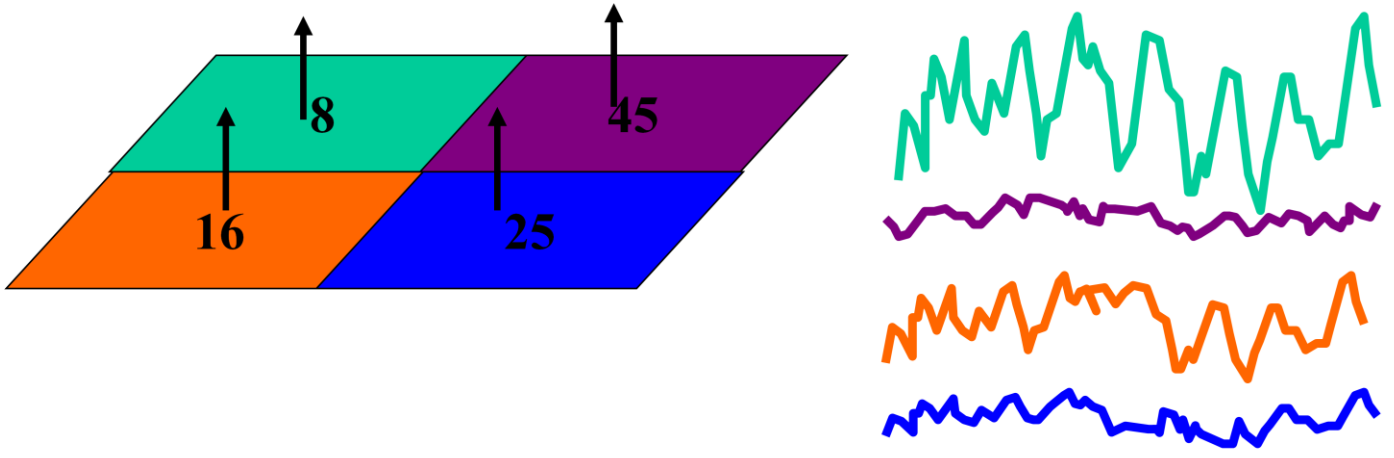


Next MIP Level – same thing

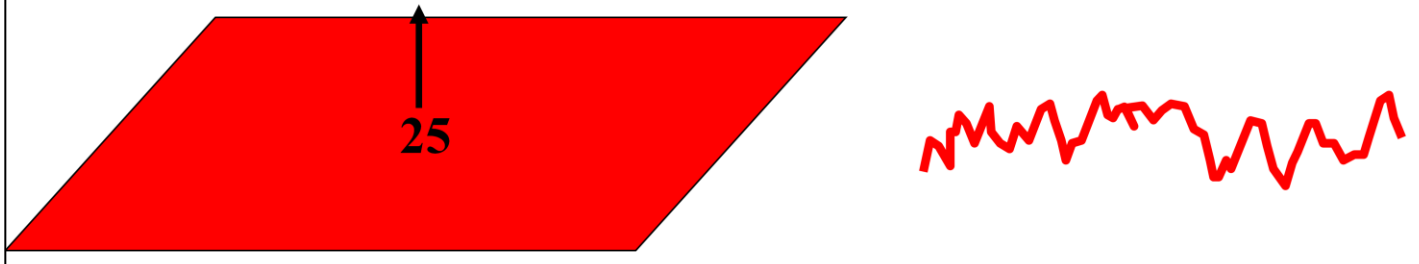


If the above material set of pixels represent a microfacet distribution, does the average represent the same set of microfacets? Here, yes because the sample points all ohave the same # of microfacets

Examining a mip level

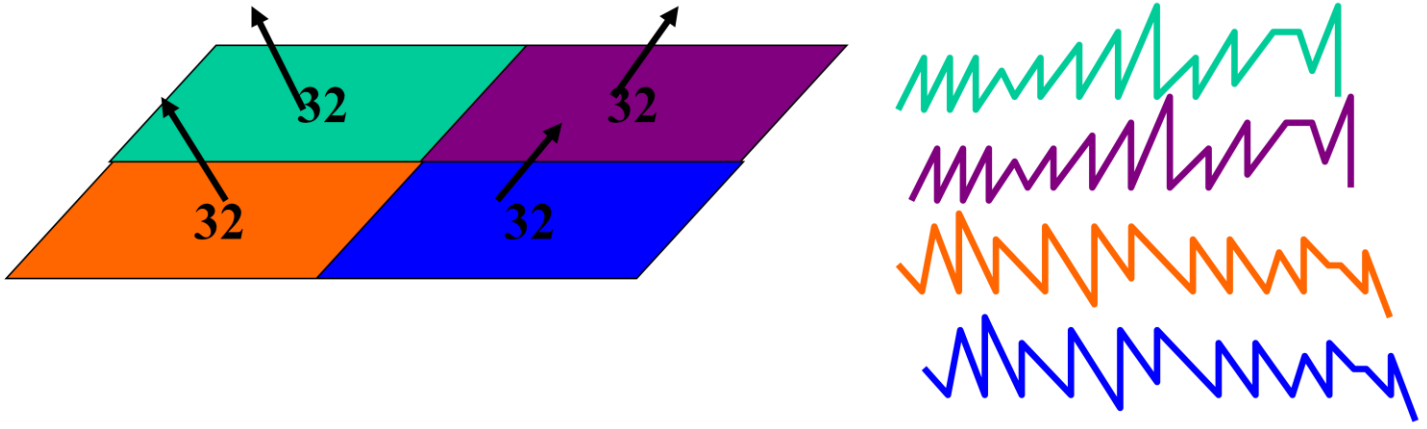


Next MIP Level, Average Power?

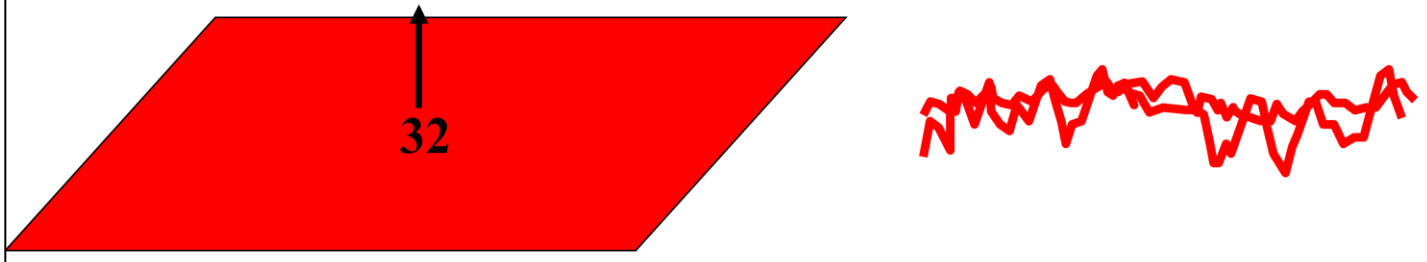


If the above material set of pixels represent a microfacet distribution, does the average represent the same set of microfacets? No.

Examining a mip level

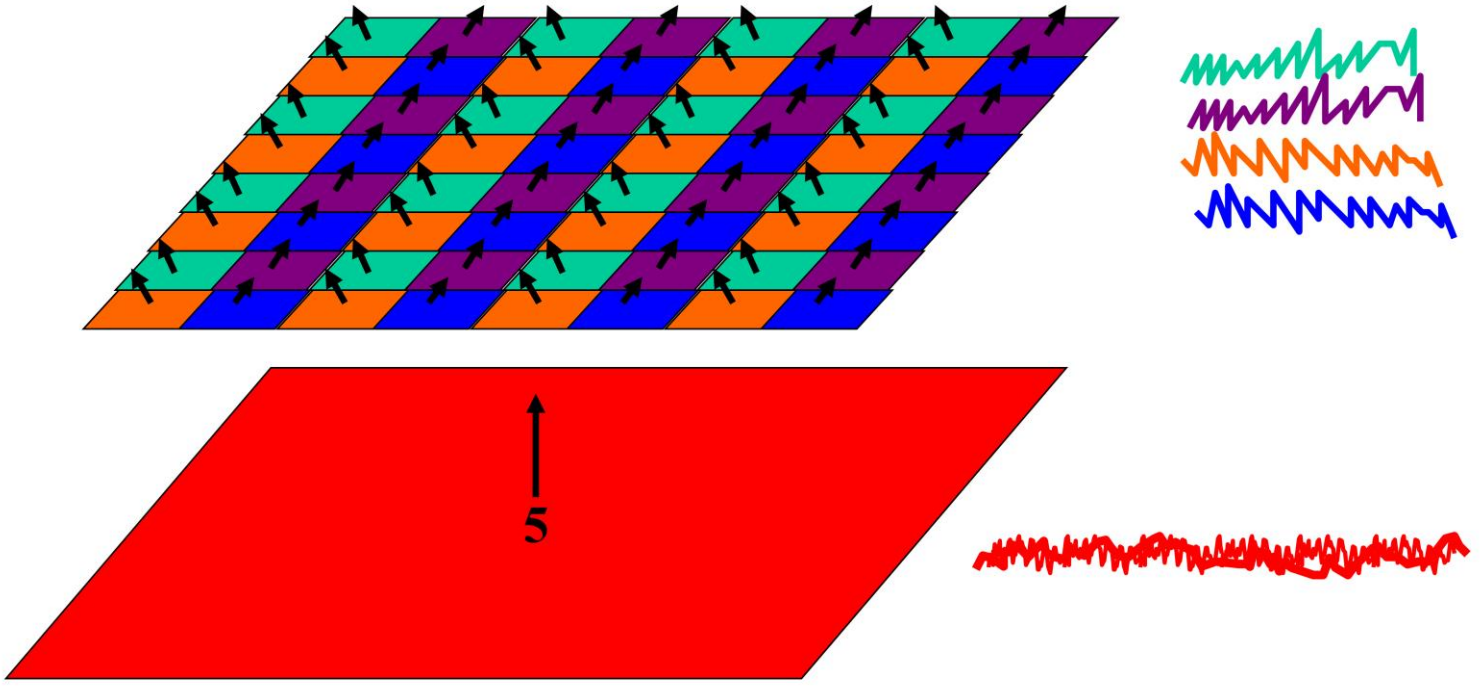


Next MIP Level, Average power, normal?



If the above material set of pixels represent a microfacet distribution, does the average represent the same set of microfacets? No.

What about a larger patch and a lower mip?



Even though all the original powers were 32, the mip level's power is closer to 5 because it must average all the microfacets

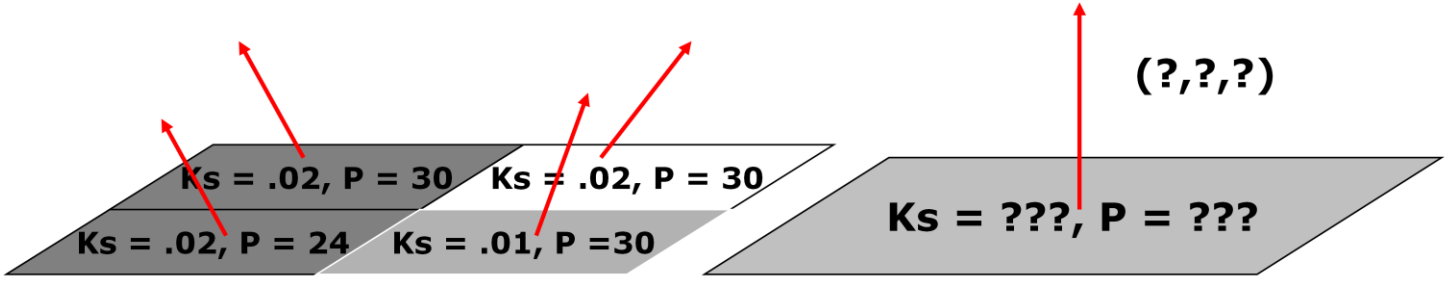
Setting up the problem

- Forgetting for a second about texture filtering, each MIP level should approximate the signal in the higher level texture
- For a BRDF model, this means that the lower MIP levels should approximate the more detailed version
- In other words, we want the BRDF at a different scale

A simple BRDF

- To start, consider the specular part of the normalized Blinn-Phong BRDF
- For each texel, there is a function F(V,L), which given a View direction and a Light direction, returns a color
- Each texel has a different N, Kd, and Power to get surface variation
- N, Ks, and Power are the parameters of a Blinn-Phong lighting model

Reflectance at a single texel



MIP Level 0 MIP Level 1

For a given light and view direction, how do we pick a Ks, a Power and Normal so that we get the same final color?

Solving for a single patch of texels

$$\frac{F_{11}(L,V)+F_{12}(L,V)+F_{21}(L,V)+F_{22}(L,V)}{4}=F(L,V)$$

For a given patch of texels w wide and h tall:

$$\frac{1}{wj}\sum_{i=0}^{w}\sum_{j=0}^{h}F_{ij}(L,V) \;=F(L,V)$$

Where,

$$F_{ij}(L,V)=p_{ij}k_{ij}(N_{ij}\cdot|L+V|)^{p_{ij}}$$

We care about all pairs of L and V

Creating an error function for a single pair of L and V:

$$e(N,k,p) = \left(\frac{1}{wj} \sum_{i=0}^{w} \sum_{j=0}^{h} F_{ij}(L,V) - F(L,V) \right)^2$$

Need to find the minimum error across all possible V and L

$$E(N,k,p) = \int_L \int_V \left(\frac{1}{wj} \sum_{i=0}^{w} \sum_{j=0}^{h} F_{ij}(L,V) - F(L,V) \right)^2$$

Creating a MIP filter

- We want to find a value of N, k, and p such that E(N, k, p) is the smallest value possible
- This is a non-linear optimization problem
- There are many ways to fit non-linear problems, for this problem we chose to use BFGS which is a quasi-Newton algorithm

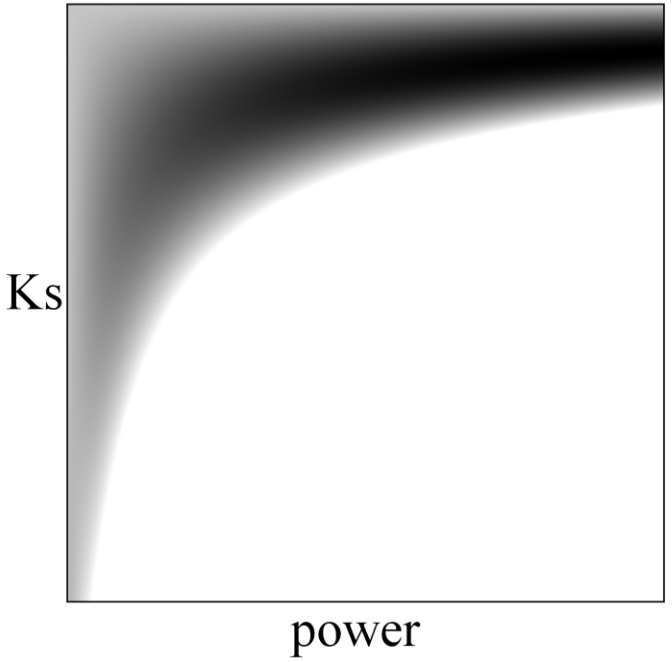
Non Linear fitting

- BFGS requires the partial derivatives of the error function with respect to each parameter we want to fit
- The derivatives must be continuous over the space we are interested in
- Double integral cannot be solved analytically, so it is numerically evaluated
- A good way to express the normal is as an X, Y with an implied Z. But error function must take care to keep the normal normalized

Giving a good first guess

- Non-linear algorithms work well when we don't have many local minima and we have a reasonable starting guess
- For the Blinn-Phong, seed of the averaged Normal, the averaged Ks, and a low power - less then half of the previous level works well
- But, power should not be near or less then 1!

Non linear with Blinn Phong



With the Blinn-Phong BRDF, the error plot shows that the function is mostly smooth. As long as we take a guess in the dark region, we should be fine. Graph is log(1 + error)

Speeding it up

- Running a BFGS with that error function took a few hours on a 512x512 texture
- Gives a good fit, but still too slow
- However, this function can be refactored in terms of the half angle
- This reduces dimensionality from 4 to 2!
- Additionally, summation in middle of integral is constant for a given H, so this can be precomputed and reused, assuming enough memory

Speeding it up

$$E(N,k,p) = \int_H \left(\frac{1}{wj} \sum_{i=0}^{w} \sum_{j=0}^{h} F_{ij}(H) - F(H) \right)^2$$

But, we must use Half Angle Distribution

Results

- Resulting MIP level looks closer to original detail
- Turns out that the optimal normal is usually pretty close to the average normal. You can get away with minimizing just P and Ks
- The power decreases rapidly for a rough surface. Each MIP has about 50-70% of the exponent of its previous level!
- An object with a shiny rough surface will appear dull as the object is further away

Dealing with Anisotropy

- Using a simple Blinn-Phong BRDF doesn't allow anisotropic effects to be captured at lower MIP levels
- A simplified, half angle version of the Ashikhmin-Shirley BRDF can do this
- We now have E(N, T, Ks, Pu, Pv), that is, we have 2 powers for the tangent frame - and the Tangent frame might vary per pixel
- Anisotropic objects, e.g. a record with grooves on it should stay anisotropic

Why Ashikhmin-Shirley ?

- Convergence is generally better, compared to other BRDFS
- Is half angle based – so MIP evaluation is still fast
- Is data light, computation heavy
- Can exactly represent Blinn-Phong

Anisotropy

- For more complex BRDFs, one should take care to ensure function well behaved at limit points

$$F(H) = K_s \sqrt{P_u P_v} (N \cdot H)^{\frac{P_u (T_u \cdot H)^2 + P_v (T_v \cdot H)^2}{(N \cdot H)^2}}$$

$$\lim_{(N \cdot H) \to 0} (N \cdot H)^{\frac{P_u (T_u \cdot H)^2 + P_v (T_v \cdot H)^2}{(N \cdot H)^2}} = e^{-\frac{1}{2} P_u (T_u \cdot H)^2 + P_v (T_v \cdot H)^2}$$

A side note:

- Artists can create extremely high resolution height maps, modeling the micro geometry aspects of a surface
- Brush strokes, grooves, scratches etc would just appear on the high-res height map
- A fitting algorithm could then capture the various aspects – roughness, anisotropy, etc.

Solution for high quality rendering

1) Determine objects maximum MIP level
2) Render from that MIP level downward, culling off unseen polygons
3) Compute lower MIP levels
4) Draw object with the lit texture, using best filter possible*
5) At lower MIP levels, once texel variation is minimal, disable TSL

**Optionally blend MIP transitions to eliminate any popping*

System considerations

- Don't need a texture for each object – just a set of textures which we recycle
- Can reduce render resolution to adjust framerate

Final thoughts on fitting

- More research to be done on BRDF fitting
- Untackled issues : more anisotropic fitting, need to look at more data
- Optimizing for bilinear reconstruction would also help direct rasterization at the expense of removing some detail

More thoughts

- Even if you don't use Texture space lighting, fitting non linear functions to create MIP maps should be done
- It's free at runtime!
- Texture Space Lighting is an easily scalable feature. It can be enabled on newer hardware
- Because TSL gets computed in point sampling mode and pasted once, we can invest a resources in a much better reconstruction filter – e.g. bicubic with better anisotropy