Federal University of Minas Gerais

Computer Science Department

Advised Project in Computer Science I

Final report

Relighting of Human Avatars for Videos

Scientific Research

Student:
Guilherme Torres <torres@dcc.ufmg.br>

Advisor:
Erickson Rangel do Nascimento <erickson@dcc.ufmg.br>

November 2019

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG

Abstract

Relighting, and by consequence, inverse rendering, have been widely explored classes of problems in computer vision and computer graphics, but yet the methods used in recent literature remain divided and, for the most part, particular for a class of objects [1, 2, 3]. In this work, we explore and research relighting methods specific for human shapes in a three-dimensional setting, and take further steps to increase the realism of a motion transfer process, which is to synthesize new videos of people in a different context where they were initially recorded, taking into account 3D shape, appearance, and motion constraints. We start by taking a precomputed radiance transfer approach, which uses spherical harmonics lighting to relight the image of the human model which is going through a motion transfer process. Three approaches are used by the authors for relighting the human shape, namely, a two-dimensional image-domain renderer, a custom OpenGL renderer and a Blender script. Finally, we conclude that relighting in 3D from inferred spherical harmonics coefficients is a feasible task and we plan on exploring it further in the next months.

Keywords: computer graphics, computer vision, illumination, relighting, inverse rendering, motion transfer, precomputed radiance transfer, spherical harmonics.

1 Introduction

The general goal of computer vision is to improve the quality of techniques for visual data manipulation and modelling techniques. Current research in artificial intelligence has been engaged in a quest to not only reach human levels at information processing and decision making, but to overcome them. In this work, we deal with examples that are on the boundary between graphics and computer vision, namely **inverse rendering and relighting**. All these terms will be strictly defined and elaborated in this work.

Rendering is to computationally produce an image given a mathematical model of a scene, namely, the set containing the geometry of the surfaces in a scene, their materials and a lighting configuration. Materials are functions which describe a surface's interaction with the light. All these factors can vary in both their computational representation and mathematical implications, that is, a geometry, for instance, can be represented as a polygon mesh, a NURBS surface, voxels, among many other possibilities.

Given the above definition of rendering, it is intuitive to picture what inverse rendering means. Given an image, the goal is to infer data about its geometry, materials, lighting, or both. Inverse rendering has been traditionally tackled as an optimization problem, or with regression. Various examples for many cases were presented by Marschner [4].

Inverse rendering is necessary for relighting [5]. Relighting is the rendering of an alternative image with a different lighting configuration given its materials and geometry. That means, an inverse rendering process with relighting as a goal should not only find the lighting configuration of the object being relighted, but the materials so that we can have the object potentially unshaded.

Applications of this work include, but are not limited to, improvement of augmented reality, image editing, video game applications, forensic science, cinema industry and special video effects. In addition, the development of studies related to illumination on computer graphics might lead to a general improvement in the representation of plausible (realistic) images and potentially open up possibilities for real-time applications.

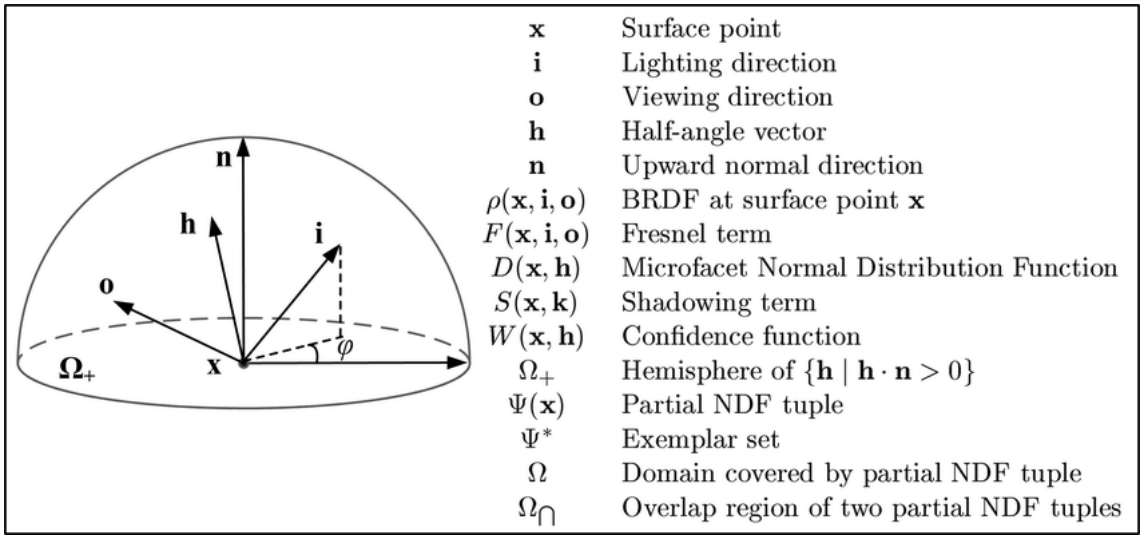
Another application of a development in relighting studies is to increase the realism of **motion transfer** process. Motion transfer is to synthesize new videos of people in a different context where they were initially recorded, taking into account 3D shape, appearance, and motion constraints. [6]. Motion transfer can be used transport a human being from one setting to the other in a realistic way, and thus evidently, making him subjected to another illumination which is not the original one. Therefore, an optimal relighting would be the ideal way to complement a motion transfer process.

Relighting methods typically rely on 2D image processing to work on [2, 5]. This is understandable, because using a per-pixel geometry configuration requires less computational effort in order to inverse render an image. But since we are working in a motion transfer pipeline, the best choice is to use the geometry of the model inferred by the motion transfer process itself, so that we will not need to inverse

render it again.

Finally, something that needs to be taken into consideration when relighting an image is the light model which is going to be used. Bui Tuong Phong's classic lighting model has been consistent for years, but learning the light as point lights with a given position and energy in a continuous space is a painful task for a neural network, which is the usual technique to inverse render an image, given the state-of-the-art CNN capabilities [5].

One alternative way to represent light is to use a model of it as a function over the hemisphere over the differential area segment defined by each fragment of the surface. This is the essence of **spherical harmonics lighting**. This lighting model was used by Sloan et al. [7] for **precomputer radiance transfer**, a rendering method where the spherical harmonics lighting function is approximated by a Monte Carlo method. This is the lighting model we use in this work, because the neural network responsible for the inverse rendering is able to easily learn a few coefficients to approximate the spherical harmonics function.



$\mathbf{x}$	Surface point
$\mathbf{i}$	Lighting direction
$\mathbf{o}$	Viewing direction
$\mathbf{h}$	Half-angle vector
$\mathbf{n}$	Upward normal direction
$\rho(\mathbf{x}, \mathbf{i}, \mathbf{o})$	BRDF at surface point $\mathbf{x}$
$F(\mathbf{x}, \mathbf{i}, \mathbf{o})$	Fresnel term
$D(\mathbf{x}, \mathbf{h})$	Microfacet Normal Distribution Function
$S(\mathbf{x}, \mathbf{k})$	Shadowing term
$W(\mathbf{x}, \mathbf{h})$	Confidence function
Ω_+	Hemisphere of $\{\mathbf{h} \mid \mathbf{h} \cdot \mathbf{n} > 0\}$
$\Psi(\mathbf{x})$	Partial NDF tuple
Ψ^*	Exemplar set
Ω	Domain covered by partial NDF tuple
$\Omega_\cap$	Overlap region of two partial NDF tuples

Figure 1: Symbols used in BRDF definition. Image from Wang et al. [9].

2 Related works and theory

This section presents some background for the research presented in this work. We start with a review of lighting models which provides an important context for a relighting work based off a lighting model.

2.1 Lighting models

Lighting models are models used to determine the light at the surface of a object in the scene. They are relevant for this work because we need a computational representation of lighting to work with.

There is a formulation for lighting on computer graphics that physically-based. It takes into account not only the light incoming from the light sources but also visibility of a point in the object's geometry and global illumination. The rendering equation is as follows [8] The BRDF components are explained at Figure 1.

$$L(x, \vec{\omega_o}) = L_e(x, \vec{\omega_o}) + \int_S f_r(x, \vec{\omega_o}, \vec{\omega_i}) L(x', \omega_i) G(x, x') V(x, x') d\omega_i, \tag{1}$$

being:

1. $L(x, \vec{\omega_o})$ the radiance of the ω_o ray coming from the x position;
2. $L_e(x, \vec{\omega_o})$ the light emitted by the surface itself;
3. $\int_S d\omega_i$ the sum of all the rays of light being received on the hemisphere defined by the surface point x;
4. $f_r(x, \vec{\omega_o}, \vec{\omega_i})$ the material (also referred to as BRDF, or **Bidirectional Reflectance Distribution Function**) which receives an incoming ray ω_i and reflects it to the ω_o direction;

5. $L(x', \omega_i)$ the light from position x' arriving from another light emitter along ω_i;

6. $G(x, x')$ the geometric relation between x and x'. In simpler models, this is just reduced to a dot product for instance;

7. Finally, $V(x, x')$ is a binary visibility test, which is 1 in case a ray can travel from x' to x without being interrupted or zero otherwise.

This equation involves global illumination factors, and it is an extremely hard task to simulate this with precision in real time. Fortunately, the lighting models used in real time rendering are simplifications of this model, and yet they can lead to fairly realistic results, as we'll see later on.

2.1.1 Phong's lighting model

The classical lighting model for computer graphics was presented by Bui Tuong Phong in his 1975 paper [10], and it's usually referred to as *dot product lighting* [8], because it is essentially a dot product between the light incident ray and the normal of the surface at a given point. This is the general equation of Phong's lighting model:

$$I_{out} = I_d + I_s + I_a = \frac{\rho}{\pi}(\vec{\omega_i} \cdot \vec{n}) + k_s(\vec{\omega_r} \cdot \vec{n})^r + k_a\rho_a, \tag{2}$$

where:

1. I_{out} is the output color;

2. ρ is the albedo of the material;

3. $\vec{\omega_i}$ is the incident direction of light. This is a unit vector, therefore, the product between this vector and another unit vector is equal to the cosine of the angle defined by them. For now on we'll use this notation specifically to denote the incident ray of light throughout this document;

4. $\vec{n}$ is the normal of the surface;

5. k_s is the specular light coefficient;

6. $\vec{\omega_r}$ is the reflected ray of light on the surface, which is given by $2(\vec{\omega_i} \cdot \vec{n})\vec{n} - \vec{\omega_i}$

7. r is the roughness of the material;

8. k_a is the ambient light coefficient and

9. ρ_a is the albedo of the ambient light.

A material with $k_s = 0$ is also referred to as a **Lambertian material**. In this work, for means of simplification, we'll assume that the human skin is Lambertian, therefore, it reflects light equally to all directions.

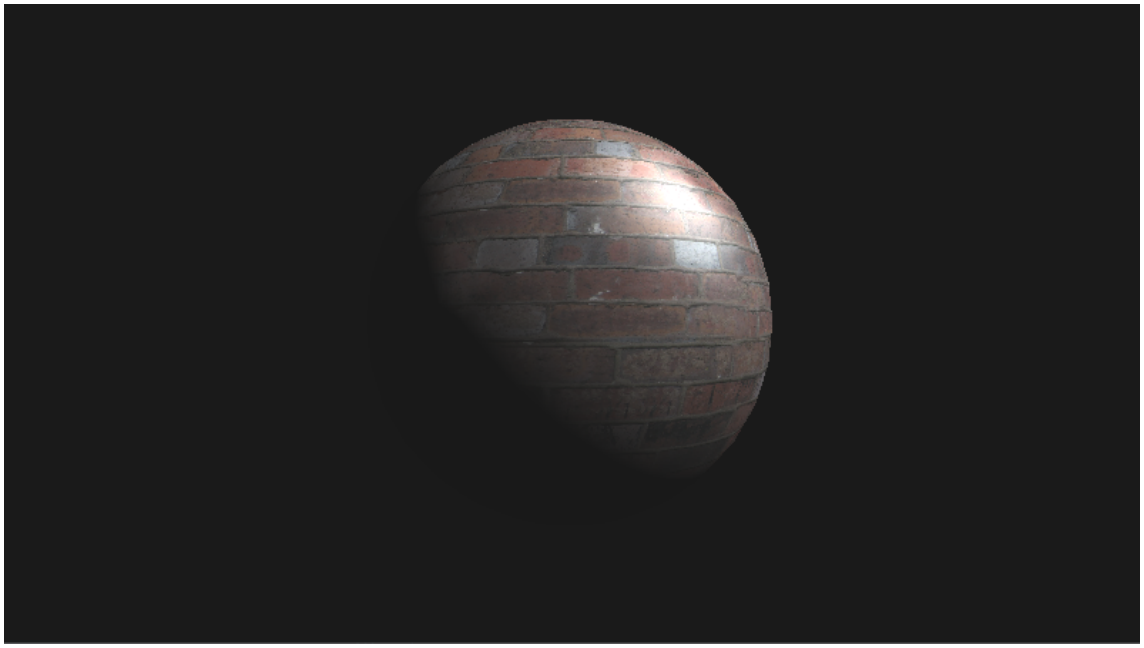


Figure 2: Sphere with bricks texture and a simple Phong BRDF material with a point light.

Phong's lighting model is a simple way of representing light interactions, but yet it has some realistic value, as we can see in Figure 2. That realism can be eventually incrased by using textures, for instance, bump maps. In Figure 3, there is an example of the same sphere and lighting configuration with a **normal map** (a texture with normal offsets per pixel), which makes it a bit more realistic on sight.

One issue with Phong's lighting model is that the ambient light is not physically plausible, because it assumes that the surface material may reflect more light than it receives. On the other hand, Phong's BRDF is still widely used due to its simplicity.

2.1.2 Local illumination model

The classic local light model defines two types of light: a point light (which has a position, a color and maybe an energy modifier), which sends rays of lights equally to all directions (see Figure 4), and a directional light, which only sends light rays in one direction (see Figure 5; it can be considered as a point light which is so far away form the target surface, the rays of light become parallel to each other).

This way of representing the light is typically combined with Phong's lighting model to produce images, since it's pretty much feasible in real time and it provides the right parameters for the $\vec{\omega_i}$ in Phong's equation.

Even though it is a simple model and is rather efficient at simulating light sources like light bulbs, this model lacks the proper treatment of global illumination. Which means that there is no way to properly simulate the rays coming out of an emitting object which has a volume with this type of representation. In addition, we should keep in mind that we're doing inverse rendering using a neural network, and as said previously, trying to estimate the energy, color, amount and position of each light, also considering the differences between a point light and a directional light in a

Figure 3: Same Phong BRDF material with a normal map. Renderer can be found at https://github.com/torresguilherme/bits-of-opengl

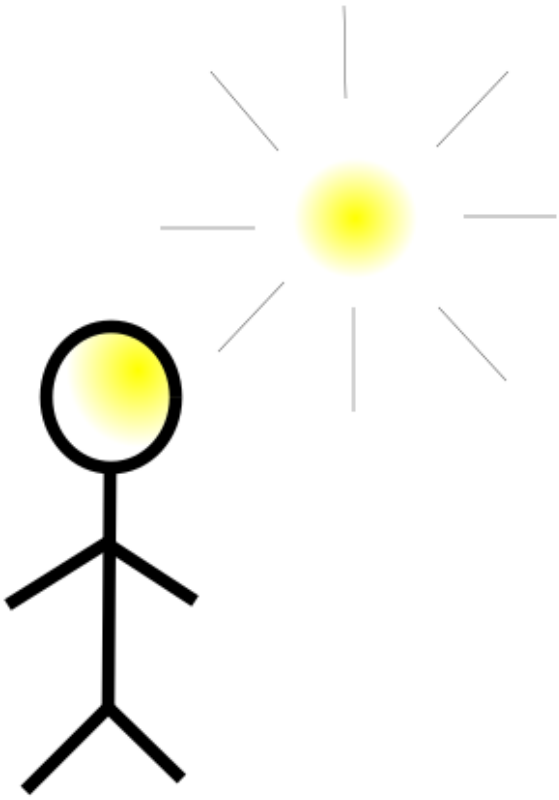


Figure 4: Point light model illustrated. Notice how it sends light equally to all directions.

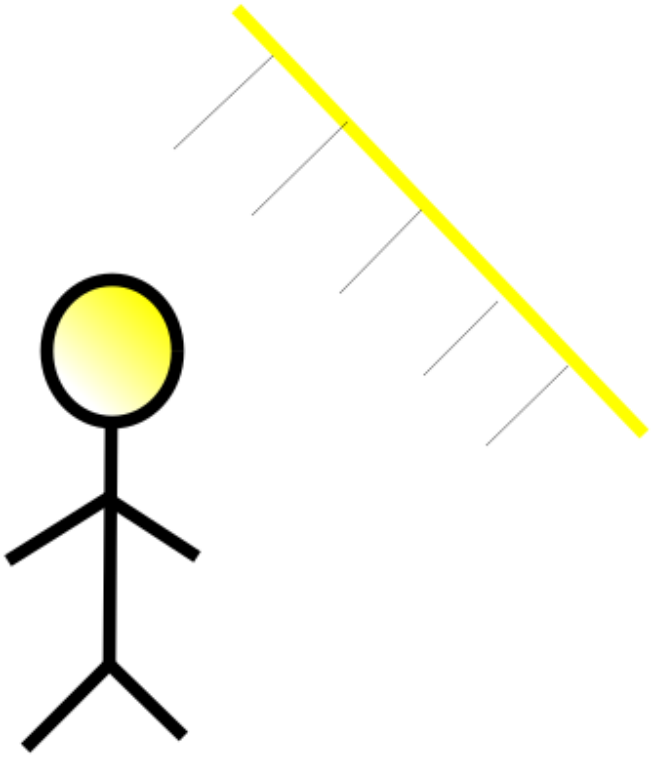


Figure 5: Directional light model illustrated.

scene is a very tough task.

Fortunately, in the early 2000s a new light model was introduced in the SIGGRAPH paper by Sloan et al. [7], called the **spherical harmonics**.

2.1.3 Spherical harmonics

Spherical harmonics have had their application in computer graphics since Sloan et al. [7] published their paper. The essence of the technique is an approximation of an ambient light function, which is precomputed. This lighting function is decomposed in bands, in an analogous way to the Fourier transforms, but in a two-dimensional space instead of one-dimensional, as we can see in Figure 6.

A harmonic is a decomposition in associated Legendre polynomials, which can be calculated according to recurrence equations. The equations are defined as:

$$\begin{cases} (l-m)P_l^m = x(2l-1)P_{l-1}^m - (l-m+1)P_{l-2}^m \\ P_m^m = (-1)^m(2m-1)!!(1-x^2)^{\frac{m}{2}} \\ P_{m+1}^m = x(2m+1)P_m^m \end{cases} \tag{3}$$

where l is the band index, m is an integer modifier that can vary between $[0, l]$ and $n!!$ is the **double-factorial** (product of all odd numbers from 1 to n). These Legendre polynomials are defined between $[-1, 1]$.

The second rule doesn't depend on other rules, so it's the ideal one to start the

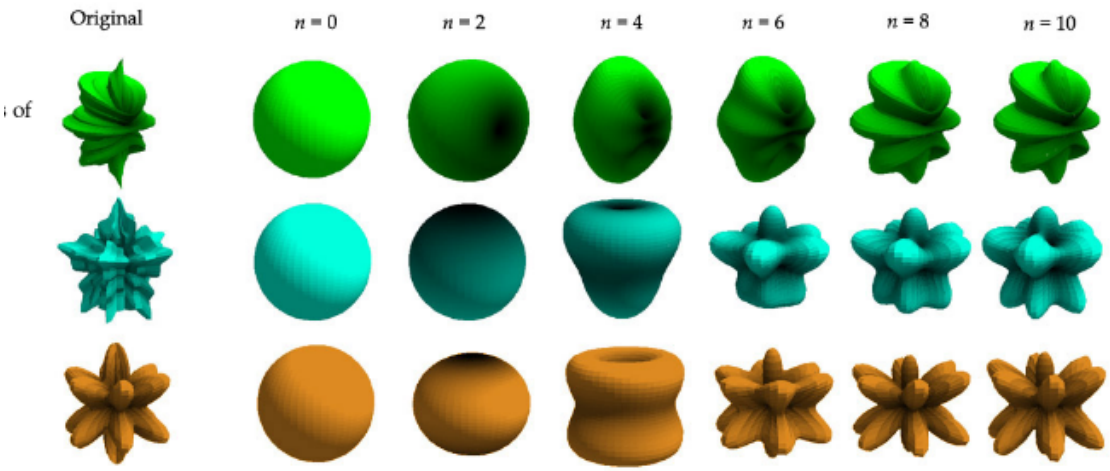


Figure 6: Spherical harmonics approximating a 2D function with an increasing number of bands. Image from Green's report on spherical harmonics lighting [8].

process. The third one allows us to lift a band higher and the first allows us to obtain a new frequency by combining two lower bands.

In order to decompose a two-dimensional function, we use spherical coordinates. Thus the spherical harmonics are usually represented as:

$$y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi) P_l^m \cos\theta, & m > 0 \\ \sqrt{2}K_l^m \sin(-m\phi) P_l^{-m} \cos\theta, & m < 0 \\ K_l^0 P_l^0 \cos\theta, & m = 0 \end{cases}, \tag{4}$$

where K_l^m is just a normalization factor, to make sure that $y_l^m(\theta,\phi)$ will be a number in the interval $[-1,1]$.

$$K_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}. \tag{5}$$

In order to project a lighting function into spherical harmonics, we just need to multiply them by the y_l^m value given a direction and integrate:

$$c_l^m = \int_S f(\vec{\omega_i}) y_l^m(\vec{\omega_i}) d\vec{\omega_i}. \tag{6}$$

Analytically integrating over a two-dimensional function is a very costly process and arguably unfeasible in shader code. So, in order to approximate this function, we multiply our spherical harmonics functions by **random samples $\vec{\omega_i}$ iterated over the hemisphere over the differential area of each fragment of the surface**, as a Monte Carlo approximation:

$$f(\vec{\omega_i}) \approx \sum_{l=0}^{n-1} \sum_{m=-1}^{l} c_l^m y_l^m = \sum_{i=0}^{n^2} c_i y_i, \tag{7}$$

given that n is the number of coefficients we have to approximate the function. We need l^2 coefficients because of that (as one can see, n is the square of the approximation band).

2.2 Precomputed radiance transfer

Precomputed radiance transfer (PRT) is a technique which consists of, as the name suggests, precomputing how a static object will reflect the incident light, or one could say, precomputing the colors that will be shown on the screen. This allows us to simulate more complex models of ambient light then the Phong model, namely a spherical harmonics lighting (further referred to as SH).

Although the general idea seems simple, in order to generate PRT images, one needs to be familiar with concepts such as projection into basis functions, empirical function integration and light transport theory [11]. Here we try to present the general idea of the algorithm.

Suppose we have a light distribution function $f(x, \vec{\omega_i})$. What we need to find in order to simulate the SH lighting is the projection of the function over SH bands. The way we'll do this depends on how the light is represented in our memory. The most common case is as shown by Slomp et al. [11], where a light probe serves as an environment map. In this case, this light probe needs to be wrapped around the object as a texture and it will contain the value of $f(x, \vec{\omega_i})$. This is mostly done in game engines as the light probes are easy for the user to edit for custom ambient lighting. So, in order to find the light coefficient at a given point, given a random sample $\vec{\omega_i}$, one can approximate equation (6) with a Monte Carlo integration:

$$c_k = \frac{4\pi}{n} \sum_{j=0}^{l^2-1} f(x, \vec{\omega_j}) y_k(\vec{\omega_j}), \tag{8}$$

in which $\frac{4\pi}{n}$ is the inverse of the probability distribution function of the sampling over the hemisphere, divided by the n number of samples, and l is the SH band.

The other scenario is what happens in Kanamori and Endo's method [5], which returns a l^2 number of spherical harmonics light coefficients for an image environment. Therefore, given the coefficient vector L for one color channel, the value of the color channel can be computed as a dot product, which follows the same Monte Carlo integration principle;

$$c_k = \frac{4\pi}{n} \sum_{i=0}^{n-1} \sum_{j=0}^{l^2-1} T_{ij} L_j y_j(\vec{\omega_i}), \forall 0 < k < K, \tag{9}$$

being K the total number of fragments (pixels) in the image, and T is a **transport matrix** (a matrix which defines the influence of the geometry in the rendering at a local point). Equation (9) is very important for our work because it is ultimately what the rendered is doing in order to compute the light.

2.2.1 Transport matrices for a Lambertian material

If our material is Lambertian, that means, if it reflects light equally to all directions given a point and the hemisphere defined by the area over it, then what happens is that the transport matrix is nothing short of the dot product between the incoming light ray and the normal of the surface at a given point. Or, as we can illustrate mathematically, this is the rendering equation for a Lambertian material:

$$c_{LB}(x) = \frac{\rho_x}{\pi} \int_S L(x, \vec{\omega_i}) \max(\vec{n_x} \cdot \vec{\omega_i}, 0) d\vec{\omega_i}. \tag{10}$$

Notice that this equation is an elaboration of the Phong lighting model (2) without the specular component. Therefore, the transport matrix can be generalized as:

$$T_{ij} = \frac{\rho_x}{\pi} \max(\vec{n_x} \cdot \vec{\omega_i}, 0). \tag{11}$$

2.2.2 Transport matrices with shadowing

When adding shadowing to the equation, we just use the visibility test from (1) $V(x', x)$. That can be done in a shader using ray casting. This is how the transport matrix is defined with a visibility test:

$$T_{ij} = \frac{\rho_x}{\pi} V(x, \vec{\omega_i}) \max(\vec{n_x} \cdot \vec{\omega_i}, 0). \tag{12}$$

2.2.3 Global illumination

Finally, we add global illumination to the equation. That means that the ambient light will depend not only on the light given by the environment map, but also the part that's being reflected by the objects themselves. Expressing it as an integral, we can write:

$$c_{GI}(x) = c_{LB}(x) + \frac{\rho_x}{\pi} \int_S L_{GI}(x', \vec{\omega_i})(1 - V(x, \vec{\omega_i})) \max(\vec{n_x} \cdot \vec{\omega_i}, 0) d\vec{\omega_i}. \tag{13}$$

Notice how it depends on the c_{LB} result from the previous sections. Also we have in this equation the following terms being considered:

1. $L_{GI}(x', \vec{\omega_i})$ is the light emitted by the object that intercepts the vision ray, by the x' point;
2. $V(x, \vec{\omega_i})$ is the previous section visibility test. Notice how the integral is reduced to zero if $V = 1$.

The algorithm to render the image using the global illumination factor as described by Green [8] is as follows:

1. For each shading point x on the model, calculate the direct lighting transfer function at that point ($c_{LB}(x)$ for instance).

2. Fire random ray samples from the point and see if they hit a location. If yes, linearly interpolate the SH of each vertex on the point on the triangle where the ray has hit.

3. Multiply these values by the dot product between the x point and the $\vec{\omega_i}$ sample, and divide by the PDF as it's done in a Monte Carlo integration.

4. Repeat from x' to a new x'' until a set constant B number of bounces has been hit, or until no energy is transferred from an object to another.

This will give you the $L_{GI}(x', \vec{\omega_i})$ factor we previously covered and will be enough to simulate a decent global illumination.

Some programs like Blender already have PRT set up in a way that users can use this global illumination within their works using a Python API. This was done by Lv et al. [12] in order to generate random human datasets from computer vision tasks, for instance.

2.3 Relighting-focused inverse rendering

There is a niche of inverse rendering methods that focus on relighting. For instance, Nestmeyer et al. [1] recently developed a method that works specifically for human faces, decomposing the image into structures. Sun et al. [3] trained a neural network in order to decode an environment map and relight also a human face. Ren et al. [2] tackled the problem with a more complete inverse rendering, for inanimate objects for the most part, decomposing the scene into light and transport matrices per pixel.

One notable phenomenon in inverse rendering, considering the works that were already cited here, is that most approaches are *ad-hoc* for one class of objects, in these cases, inanimate objects, or human faces.

Despite the fact that none of the methods cited above works with full-body images of human beings, Kanamori and Endo [5] made a convolutional neural network that is able to learn spherical harmonics coefficients and transport matrices per pixel. This approach is focused in relighting human avatars, however, this is made in the two-dimensional image domain and only for individual images, which generates inconsistency (unnatural variation in the lighting) when applied to consecutive frames of a video.

Finally, it is worth mentioning that some methods work purely on the image domain, without inferring any geometry. **Intrinsic images** is one of them. An intrinsic image decomposition is a separation or the input image I into an R reflectance (or albedo) image and an S grey-scale shading image, in such a way that $R \otimes S = I$ where $\otimes$ is the Hadamard product (element-wise matrix multiplication). Li and Snavely [13] recently developed a method which learns intrinsic images using neural networks. Similarly, Tsai et al. [14] presented an **image harmonization** method which slightly changes the color of the target given the inferred environment map from the image.

3 Methodology

3.1 Relighting in the two-dimensional image domain

The first method used to relight the human for a video was to use an approach inspired by the frame-by-frame approach from Kanamori and Endo [5]. Given the image width w, and the height h, their network offers a method for inferring the $L_{9\times3}$ light map from a scene as 9 SH coefficients, an albedo (unshaded material) $A_{w\times h\times3}$ map and per-pixel transport matrix coefficients $T_{w\times h\times9}$, in such a way that the relighted image $R = T \times L \times A$.

This method was used in 2D relighting, and it was observed that the ambient light estimation achieved good results for human shapes, but the fact that the transport matrix is also inferred from the network means that, for a motion transfer pipeline, we would need to inverse-render a human shape that is already inverse rendered. Since we have that human geometry as an SMPL shape [15], we can render with "ground-truth" geometry instead of estimating it again.

3.2 Relighting in 3D with a custom renderer

A custom PRT renderer with SH lighting was built with OpenGL 3.3. The light matrices used were from Kanamori's inverse rendering method. The transport matrices were calculated according to the surface normals in the given fragment points and their SH projections, using a Monte Carlo integration, as described in section 2. The materials were assumed to be Lambertian and uniform at all points.

This approach, although it presented a proof of concept, is hard to integrate with the existing systems and does not include global illumination yet, which requires a new approach.

3.3 Relighting in 3D with Blender

Blender is a graphics editing and rendering software that can do precomputed radiance transfer with a custom OSL (Open Shading Language) script, including global illumination. In order to send the custom light, we'll use the same light matrices, and for sending the human shape with the right texture, we can read an SMPL model from the disk.

Since the SMPL was already loaded in a previous inverse rendering method, and we're using Python, we can save the parameters as a *Pickle* binary and load them in our script.

The script used to interface with Blender's Python API was based off ReFRESH, a dataset generator by Lv et al. [12].

3.4 Light smoothening process

We can also do a smoothening process to assure that the light is consistent during time, since our focus is working with videos. This process can be done with a polynomial regression over the light parameters, but there are better ways to do regression between points, such as Hermite bases.

Once we are done with the regression, we can use an error parameter ϵ as the maximum euclidean distance from the regression in order to classify point that bypass this parameter as *outliers*. The outliers can then be dealt with individually, with a *correction*. This correction can change the light so that it fits on the regression line.

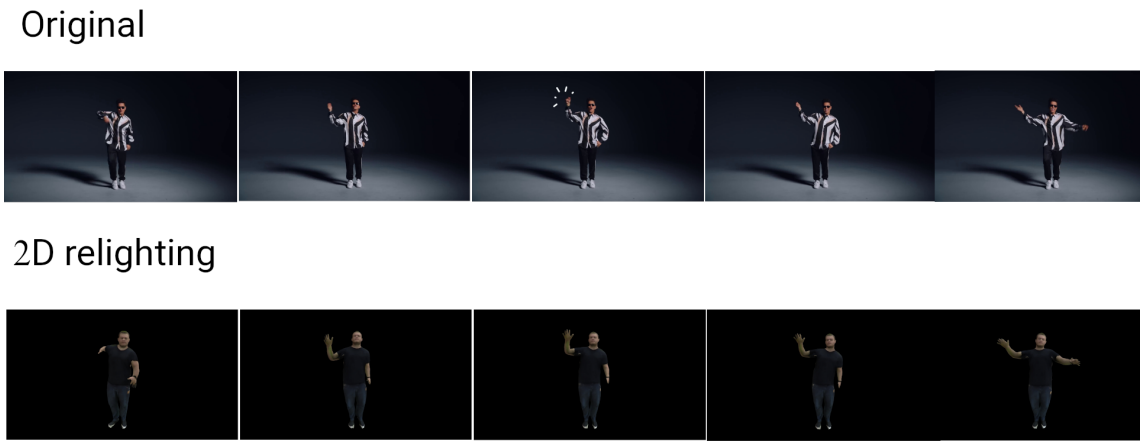


Figure 7: Kanamori and Endo's relighting with images from the Bruno Mars music video.

4 Preliminary results

Here we present the results we've got to the date. These results were obtained used non-real time rendering for all images and for all the cases, the scene which was inverse rendered were frames from Bruno Mars' music video "That's What I Like", which has a very particular light variation. The human model used for the motion transfer process and being relighted is a PhD student from our institution.

4.1 Relighting in 2D

For 2D the lighting estimation and operations on the image domain were done as described above. The multiplication operation was done in CPU and generated the results compiled in Figure 6.

4.2 Relighting in 3D with custom renderer

The custom renderer served as a proof of concept for our work. With it, the spherical harmonics lighting in 3D taking into account only the 9 SH coefficients for the light was proven to be possible.

Indeed, the custom renderer was able to simulate the color of the light and the rotations successfully, for a Utah teapot model.

Monte Carlo integration for PRT was done in the GPU, in a GLSL program. Results were compiled in Figure 7, for lights from Kanamori and Endo's training dataset. The renderer itself can be found at https://github.com/torresguilherme/spherical-harmonics-demo.

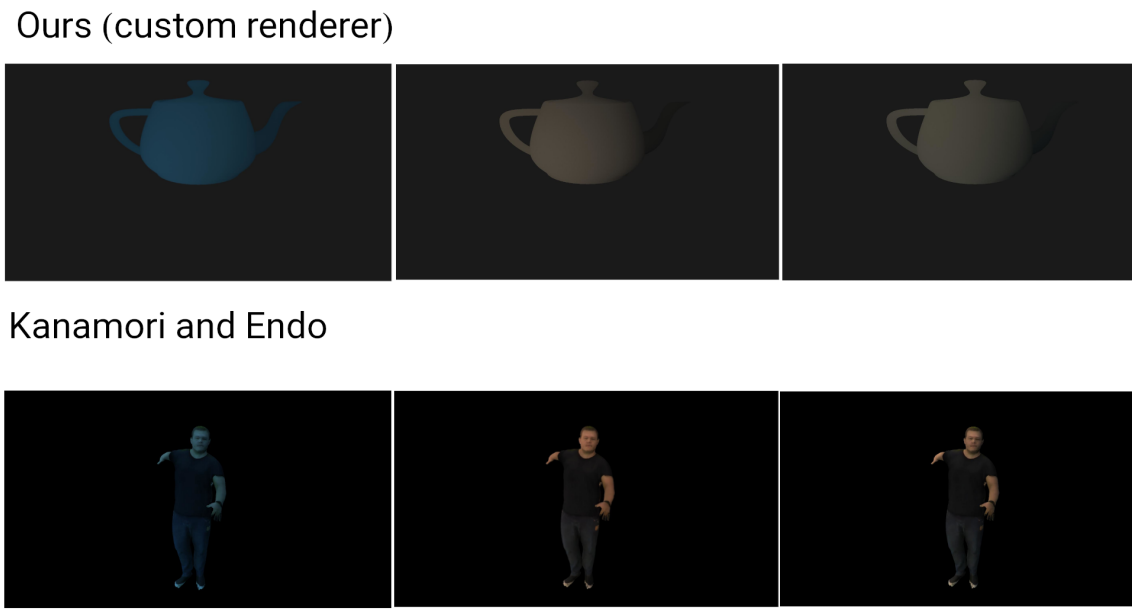


Figure 8: Custom renderer using the inferred light to render an Utah teapot.

Figure 9: Outputs from ReFRESH's `render_bodies.py` Blender script, modified to render only one human being at a given position.

4.3 Relighting with Blender

The relighting with Blender was done based on a script from ReFRESH [12]. The integration of the actual SMPL model with the rendering pipeline and the textures of the human model is yet to be done, but the light coefficients have been transferred successfully. Here we also use the lights from Kanamori and Endo's training dataset. The results we have to this date are random human textures and SMPL models in poses with the given lighting, as we see examples in Figures 8.

5 Conclusion and next steps

While arguably there are a few steps to be taken on our method, the results which were obtained to the present date have shown to be promising. Is it pretty clear that we can use a neural network for inverse rendering in order to obtain SH coefficients to do relighting, and it is also arguably true that relighting in the 3D domain with ground-truth geometry is not only possible, as shown with our 3D custom renderer, but it is the best path to be chosen in our situation.

The results in general, though they have not been compared other relighting methods on a quantitative evaluation, probably the most important baseline for our work currently is the 2D relighting from Kanamori and Endo, and the qualitative evaluation would be the most important regarding this role, given that we are trying to make a video plausible regarding human judgment.

Aside from that, the results we have produced, in general, indicate that the work is in constant development and that it's clear that the proposal is feasible under our computational and time constraints.

For the next steps, the first one is to develop the Blender interface code to make room for the actual human model used in motion transfer. That will be done by importing the SMPL parameters for the body shape, joints, positions and rotations of joints and the given inferred texture of our human model.

Furthermore, we can mention the possibility of being able to move the camera, animating its pose during the video rendering. Once the Blender interface is done, doing this will be rather straightforward.

In addition, the light smoothening is yet to be done, and there are several possible approaches for it, as it was mentioned earlier. That will be important for the making the video light incident on the human body more consistent.

The rest of the work will be concluded and submitted in the following six months.

References

[1] Thomas Nestmeyer, Iain Matthews, Jean-François Lalonde, and Andreas M. Lehrmann. Structural decompositions for end-to-end relighting. 2019.

[2] Peiran Ren, Yue Dong, Stephen Lin, Xin Tong, and Baining Guo. Image based relighting using neural networks. 2019.

[3] Tiancheng Sun, Jonathan T. Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *ACM Trans. Graph.*, 38, 4(79), 2019.

[4] Stephen Robert Marschner. *Inverse Rendering for Computer Graphics*. PhD thesis, Cornell University, 1998.

[5] Yoshihiro Kanamori and Yuki Endo. Relighting humans: Occlusion-aware inverse rendering for full-body human images. *ACM Trans. Graph.*, 37, 6(270), 2018.

[6] Wen Liu, Zhixin Piao, Jie Min, Wenhan Luo, Lin Ma, and Shenghua Gao. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. *ICCV*, 2019.

[7] Peter-Pike J. Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21, 3:527–536, 2002.

[8] Robin Green. Spherical harmonic lighting: The gritty details, 2003. Sony Computer Entertainment America.

[9] Jiaping Wang, Shuang Zhao, Xin Tong, John Snyder, and Baining Guo. Modeling anisotropic surface reflectance with example-based microfacet synthesis. *ACM Trans. Graph.*, 27, 08 2008.

[10] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of ACM 18*, 6:311–317, 1975.

[11] Marcos Paulo Berteli Slomp, Manuel M. Oliveira, and Diego Inácio Patrício. A gentle introduction to precomputed radiance transfer. *RITA*, 13(2), 2006.

[12] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, James Rehg, and Jan Kautz. Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation. *ECCV*, 2018.

[13] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. *CVPR*, 2018.

[14] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Xin Lu, and Ming-Hsuan Yang. Deep image harmonization. 2017.

[15] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.