BRDF approximation and estimation for Augmented Reality

Patrick Kühtreiber*

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria

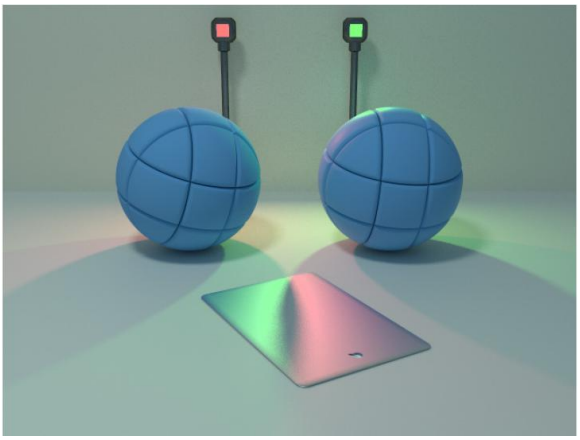

Figure 1: [Lafortune et al. 1997]

Abstract

In Augmented Reality applications it is important to have a good description of the surfaces of real objects if a consistent shading between real and virtual object is required. This thesis emphasizes on this topic as the thesis is a part of the RESHADE[1] project whose aim is to deliver a scene of virtual and real objects mixed together where difference is not noticable for the viewer. That means that virtual objects also influence the appearance of the real objects.
If such a description of a surface is not available it has to be estimated or approximated during runtime.
In my bachelor thesis I will present certain methods that deal with real-time bi-directional reflectance distribution function (BRDF) approximation in augmented reality. Of course the most important thing is that the applications I present all work in real-time and compute good (and real) looking results.
There are different methods on how to achieve this goal. Most of the methods I am going to present work via image based lighting and require a 3D polygonal mesh representation of the object whose BRDF shall be approximated. Some methods estimate the BRDF parameters via error values and provide results at each iteration.

Keywords: Real-time BRDF approximation, Augmented Reality

1 Structure of this thesis

After a brief introduction about augmented reality, BRDF and BRDF approximation I will describe the papers listed in Section 2 in detail, give a summary about the parts that concern BRDF approximation or estimation and I will conclude each method description with a short paragraph where I will point out some problems with the algorithm and their performance and usability in augmented reality. The summaries of the papers are chronologically arranged beginning with a paper from Yu et al. from 1997 and concluding with a paper from Ritschel and Grosch from 2008.
In each section I will give a summary of the presented paper with a short discussion about performance and problems which might occur. After the summary of the papers I will compare all the presented papers. The main focus here will be performance, as this is the most important part of the approximation, but photorealism (if provided) is also important.
At the end I will compare the presented methods in matter of performance and photorealism and give a general conclusion over the topic.

2 Introduction

Several papers deal with the problem of real-time BRDF approximation. BRDFs are functions with lots of parameters that describe how a certain surface reflects incoming light. If a BRDF for certain objects is not known it has to be approximated as closely as

*e-mail: patrick.kuehtreiber@chello.at
[1] http://www.cg.tuwien.ac.at/research/projects/RESHADE/

possible. In Augmented Reality applications it would be desireable to do it during runtime to allow maximal scene dynamic.
We need a good representation of the reflection behaviours of the surfaces and, what makes it even more difficult, we need them in real-time.
The most of the presented methods are rather similar, as all use image based lighting. Still there are differences in performance and photorealism which I will try to point out at the end of the section and again in the section at the end of this thesis.

The papers presented in this thesis are:

- Inverse global illumination: Recovering reflectance models of real scenes from photographs, [Yu et al. 1999]
- Image-Based Rendering of Diffuse, Specular and Glossy Surfaces from a Single Image, [Boivin and Gagalowicz 2001]
- Photorealistic rendering for augmented reality using environment illumination, [Agusanto et al. 2003]
- Recovery of material under complex illumination conditions, [Wu et al. 2004]
- A Framework for Automatically Recovering Object Shape, Reflectance and Light Sources from Calibrated Images, [Mercier et al. 2007]
- Recovering surface reflectance and multiple light locations and intensities from image data, [Xu and Wallace 2008]
- On-line estimation of diffuse materials, [Ritschel and Grosch 2008]

2.1 A survey to Augmented Reality

Basically Augmented Reality is an alteration of Virtual Reality. In Augmented Reality virtual three dimensional objects are being integrated into a real scene, so the user can still see and move in the world around him. According to Azuma Augmented Reality systems must meet the following charactaristics: [Azuma 1997]

1. Combines real and virtual
2. Interactive in real-time
3. Registered in 3D

What is important here is that all virtual objects must be interactive 3D objects. So simple blending of 2D objects does not count as Augmented Reality. Augmented Reality applications are found in medicine, entertainment and visualisation amongst others.

Figure 2: A video see through HMD. Picture from http://www.vrlogic.com/

How is this being achieved?

There are two different approaches to Augmented Reality namely optical and video. The optical approach is realized with a see-through HMD (Head-Mounted Displays). As the name says, the user sees the real world through this device but also virtual objects can be added to the users sight by reflecting these virtual images onto the combiners. While in the 90s those devices were big and not really handy, there are nowadays HMD which have the size of normal glasses. They can be adapted to smartphones which could process complex AR applications in future as there are already some rudimentary applications like World Wide Signpost[2] which lets you annotate buildings at different viewpoints.

The other possibility is to use video-see-through HMD (see Figure 2). The real world gets recorded by a camera, merged with virtual objects and then is shown on an HMD-device [Azuma 1997]. This method is nowadays used in many smartphone and tablet PC applications like games or architecture programs.

2.2 BRDF approximation

Bidirectional Reflectance Distribution Functions describe how incoming light gets reflected by a particular material and has first been described by Nicodemus et al. [Nicodemus et al. 1977]. It should at least combine both the diffuse and the specular part (i.e. highlights) of the reflection. The BRDF shown here is a 6 dimensional (location, four angles and wavelength) function, but it can be more- or less-dimensional:

$$f_r = \frac{L_r(\omega_r)}{L_i(\omega_i)\cos\theta_i d\omega i}, \tag{1}$$

where $L(\omega_r)$ marks the flux of reflected light at direction ω_r, $L(\omega_i)$ is the flux of incoming light at direction ω_i and θ_i is the angle between the surface normal at a point i and the light incident on that point. An image of the BRDF and those parameters can be seen in Figure 3.

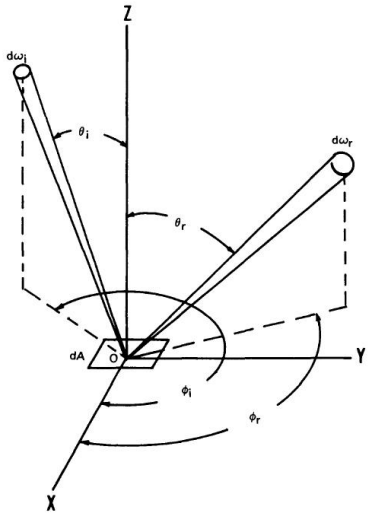

Figure 3: Geometry of incoming and reflected light. The z-axis marks the surface normal [Nicodemus et al. 1977].

[2] http://studierstube.icg.tu-graz.ac.at/handheld_ar/wwsignpost.php

Reflected radiance is defined by

$$L_r(\theta_r,\phi_r) = \int_0^{2\pi}\int_0^{\pi/2} L_i(\theta_i,\phi_i) f_r(\theta_i,\phi_i;\theta_r,\phi_r)\cos\theta_i \sin\theta_i d\theta_i d\phi_i. \quad (2)$$

where f_r is the BRDF (the rest of the parameters will be explained later in Section 9 at Equation 39 which is basically the same equation with a different parametrization) defined by two incident and two reflected angles. f_r is bi-directional because the incident and reflected directions can be reversed and the function will return the same result [Ward 1992].
If the BRDF is not known it has to be approximated or interpolated respectively. In most of these cases we use the algorithms presented by Phong [Phong 1975] or Ward [Ward 1992]. We will later see that there exist some BRDF-approximations for Augmented Reality which deliver parameters for these BRDF models. For a better approximation of the BRDF Lafortune [Lafortune et al. 1997] made an important contribution in presenting the Generalized Cosine Lobe Model, where reflectance properties at different camera angles were rendered properly with just linear basis functions. Their method works very well on broad and glossy reflectance lobes.
Basically the BRDF is the most important part of the Rendering Equation (see Equation 2). Sometimes the BRDF alone is called Rendering Equation.

2.2.1 Lambert's Law

A Lambertian reflector is a surface which can be thinked of as purely diffuse. The reflected radiant light energy from any point on the surface is calculated with Lambert's cosine law which states that the amount of radiant energy coming from any small surface area dA in a direction ϕ_N relative to the surface normal is porportional to $\cos\phi_n$. The intensity is the same for all viewing directions which can be described as

$$\alpha \frac{\cos\phi_N}{dA\cos\phi_N} = \text{constant} \quad (3)$$

where α is the viewing angle [Hearn and Baker 2004]. This constant parameter is usually spoken of as k_d which denotes the diffuse reflectance parameter.

2.2.2 Phong's reflectance model

One of the most efficient and frequently used BRDF models was introduced by Phong [Phong 1975]. He defines shading as a function which yields the intensity value of each point on the surface of an object from the characteristics of the light source, the object (the material properties) and the position of the observer (important for specular highlights) [Phong 1975].
The specular part of the reflected light at point p can be computed as

$$I_p = k_s \cos^n\phi \quad (4)$$

where ϕ is the viewing angle relative to the specular reflection direction ω_r, k_s is the specular reflectance parameter, and n is the specular exponent where the value for n is direct proportional to the specular reflectance of the surface [Phong 1975] [Hearn and Baker 2004].
The interesting part of his work is of course the specular reflectance. To simplify this model, $\cos\phi$ from the original fomula can be replaced with $V\cdot\omega_r$, where $\omega_r = (2N\cdot\omega_i)N - \omega_i$, N is the surface normal of this point, V is the viewing vector and ω_i is the direction of the incoming light [Hearn and Baker 2004].
This simplification works under the assumptions that the light source and the observer are located at infinity [Phong 1975]. That leaves us with

$$S_{specular} = k_s(V\cdot\omega_r)^n. \quad (5)$$

So for the Phong shading model you only need three parameters: the diffuse reflectance k_d, the specular reflectance k_s and the shininess n.

2.2.3 Ward's reflectance model

Here more parameters are needed as this model is more physically plausible.

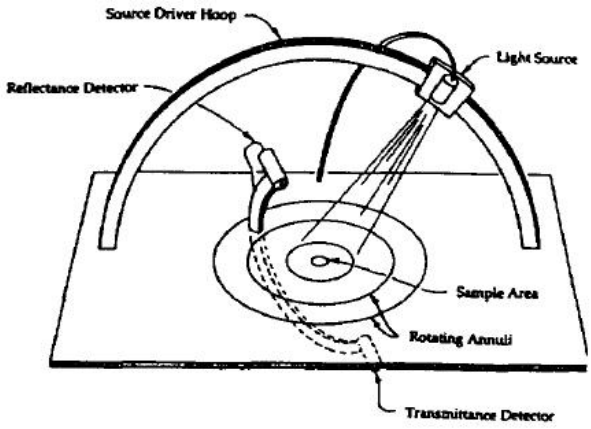

Figure 4: A gonioreflectometer with movable lightsource. Picture by [Ward 1992]

To find the correct BRDF Ward proposed to use a goniorefelctometer (see Figure 4) to get all the parameters needed and then use them for the *isotropic gaussian model*

$$f_{r,iso}(\theta_i,\phi_i;\theta_r,\phi_r) = \frac{\rho_d}{\pi} + \frac{\rho_s}{\sqrt{\cos\theta_i\cos\theta_r}}\frac{\exp[-\tan^2\delta/\alpha^2]}{4\pi\alpha^2}. \quad (6)$$

where ρ_d is the diffuse, ρ_s is the specular reflectance and α is the standard deviation of the surface slope. $\frac{1}{4\pi\alpha^2}$ is the normalization factor which replaces the geometric attenuation factors which are usually difficult to integrate [Ward 1992].
Ward states that this model is superior to the model presented by Phong (Section 2.2.2) as it is more physical plausible and the normalization factor guarantees energy balance. The alteration of the isotropic gaussian model to the anisotropic case is realtively simple. The *anisotropic gaussian model* is

$$f_r(\theta_i,\phi_i;\theta_r,\phi_r) = \frac{\rho_d}{\pi} + \frac{\rho_s}{\sqrt{\cos\theta_i\cos\theta_r}}\frac{\exp[-\tan^2\delta(\cos^2\phi/\alpha_x^2+\sin^2\phi/\alpha_y^2)]}{4\pi\alpha_x\alpha_y} \quad (7)$$

where ϕ is the azimuth angle of the half vector projected into the surface plane, α_x is the standard deviation of the surface slop in the $\vec{x}$ direction and α_y is the same for the $\vec{y}$ direction. [Ward 1992].
The papers I will present that try to approximate the parameters for Ward's reflectance model try to find all the missing parameters for this *gaussian anisotropic model*.

2.3 Finding Illumination from an image

Some of the following papers determine the scene illumination from a high dynamic range (HDR) image. This method of image based rendering was developed by Debevec [Debevec 1998] and [Debevec and Malik 1997].
Before these contributions light sources had to be determined manually or the user at least had to refine a lot of the resulting data. With the method of the image based model an HDR fisheye picture of the scene is taken and all light sources of this scene can be determined. To achieve this, they split the scene into three components:

1. Distant scene
2. Local scene
3. Virtual objects in the scene

The distant scene illuminates the objects but all light reflected to the distance will be ignored. The local scene are all the polygons and meshes that interact with the object. So here the intereflections between the objects and the scene should be taken into account.
Ideally all these three components are well modeled and positioned and a global illumination algorithm renders the whole scene.

3 Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs

An approach at image based rendering has been made by Yu et al. [Yu et al. 1999]. Although their paper has not much to do with augmented reality (and nothing with real-time) it still presents an interesting algorithm on how to gain reflectance properties from a real scene and use them afterwards as paremeters for a virtual scene, which of course can be adapted to augmented reality as they write in their introduction. They don't say that their approach works at real-time, however this paper is from 1999 and it is a good guess that it will work interactively today. Little disadvantages are that their algorithm limits itself to low parameter reflectance models, it presumes the diffuse part of the reflectance to vary arbitrarily over any surface and furthermore it presumes the specular part to be constant. Nevertheless I will discuss their contribution in this section as their sampling of diffuse and specular reflectance is rather interesting for approximating BRDFs in augmented reality applications.

Basically their algorithm creates a 3D geometric representation of the scene using a sparse set of photographs. From this rather small set it is impossible to gain all the needed information on the BRDFs, that's why they made the aforementioned limitations.
The inputs for their algorithm are the geometric model of the scene, a set of radiance maps taken from the photographs under known illumination and partitioning of the scene into areas with similar reflectance properties for the non-diffuse parts. They address the problem of gaining the illumination parameters as *inverse global illumination* and the recovering of the diffuse reflectance *inverse radiosity* [Yu et al. 1999].

Before addressing the problem of general surfaces they make a number of calculations for special cases. At first they make the assumption that all surfaces are purely diffuse reflectant.
The inverse radiosity equation is easy to calculate. For "normal" radiosity they have the well known equation as

$$B_i = E_i + \rho_i \sum_j B_j F_{ij}, \tag{8}$$

where B_i is the resulting radiosity, E_i is the emission of the light coming from that patch i, ρ_i is the diffuse albedo and F_{ij} is the form factor between the patches i and j [Sillion and Puech 1994].
From the photographs which are also taken from the lightsources we can get the radiance B_i via HDR-techniques contributed by Debevec [Debevec and Malik 1997]. Since they assumed the surfaces to be perfectly diffuse, E_i is also known. The formfactors can be retrieved from the known geometry, so that leaves us with

$$\rho_i = \frac{B_i - E_i}{\sum_j B_j F_{ij}} \tag{9}$$

to get to the diffuse albedo in this special case.

Another special case helps them to calculate the BRDF. The special case here is that they assume to have a surface with uniform BRDF which is illuminated by a point light which is again photographed with a camera. As input they get the radiance L_i from the photograph of every point P_i in direction of the camera which lets them calculate I_i, the irradiance incident at that point. They use Ward's [Ward 1992] BRDF model for this case, as it only needs two parameters the diffuse term $\frac{\rho_d}{\pi}$ and the specular term $\rho_s K(\alpha, \theta)$, where ρ_d and ρ_s are the diffuse und specular reflectence properties of the surface and $K(\alpha, \theta)$ is a function, where α is the surface roughness vector (for the specular highlight) and θ is a vector of the azimuths and elevation of the incident and viewing directions (see Section 2.2.3).
So for each surface they now get

$$L_i = \left(\frac{\rho_d}{\pi} + \rho_s K(\alpha, \theta_i) \right) I_i. \tag{10}$$

As mentioned before, L_i and I_i can be measured or calculated respectively from the photographs. θ is also known, so they just need to estimate the reflectance parameters and the roughness vector, which can be done through nonlinear optimization according to Yu et al. [Yu et al. 1999].
Now I will discuss the general case of finding BRDFs in the contribution from Yu et al. We now have surfaces which can be diffuse and/or specular. To get the BRDF of such a point P_i viewed from camera C_v, we first need to get the irradiance. We already have this equation for the special perfectly diffuse case. Generally Equation 10 is

$$L_{C_v P_i} = E_{C_v P_i} + \rho_d \sum_j L_{P_i A_j} F_{P_i A_j} + \rho_s \sum_j L_{P_i A_j} K_{C_v P_i A_j}. \tag{11}$$

Again everything except for α and the reflectance terms ρ_s and ρ_d are known. To get these parameters they need a photograph of every surface under specular lighting conditions, to record the radiance coming from that patch.

The specular component of a surface is different to the direction from the viewpoint of the camera and from the direction of another surface (see Figure 5). This difference is denoted as ΔS. To calculate these specular differences the BRDF of A_j has to be known, which is not the case. So ΔS needs to be estimated which

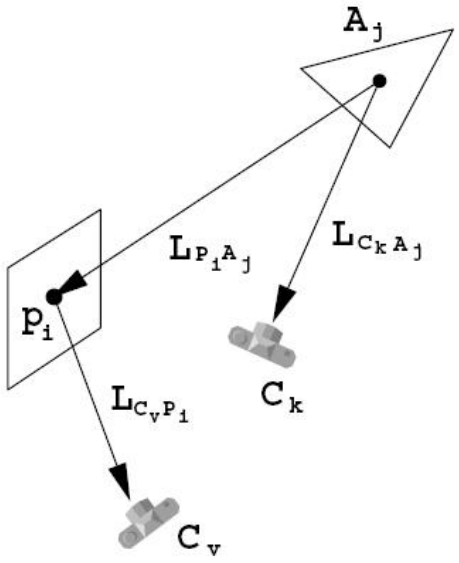

Figure 5: Specular highlight from P_i as viewed from the camera and another patch A_j. Picture from [Yu et al. 1999]

happens in an iterative framework also presented by the authors. The initial value for this iterative process sets $\Delta S = 0$, so that the L-values (the radiance) from Figure 5 are equal.

As now all the patches' radiances are recorded we have a similar optimization problem as we had above. This process can be made easier if the surfaces get subdivided into a hierarchy of patches and link the sample points. So we get the radiance from a patch to a sample point $L_{P_iA_j}$ and an associated ΔS.
Each sample point gathers radiance from not only one patch but from all the surfaces surrounding it. To obtain all the highlights the authors present the following algorithm

1. For each camera position C
2. For each polygon T
3. For each light source O
4. Obtain the intersection P between plane of T and line CO' (O' and O are symmetric about T);
5. Check if P falls inside polygon T;
6. Check if there is any occlusion between P and O;
7. Check if there is any occlusion between C and any point in a local neighborhood of P;

A highlight area is detected if P passed all the above tests.
Now to complete the inverse global illumination process they first find a BRDF for all surfaces independently. After having done this they use the found information on the radiance parameters L and the specular differences ΔS to refine the found BRDFs. They do this a couple of times until the final solution is found.

A rather huge challenge is to estimate the ΔS. Yu et al. did this with Monte Carlo raytracing with only one bounce. The bounce of a patch A_j is random but weighted by the function $K(\alpha,\theta)$ which denotes the direction of the specular light. So most of the bounces will fall into the cones Q in Figure 6, which are centered around the mirror directions. Now ΔS can be calculated as

$$\Delta S = \rho_{sA_j}(L_{Q_{P_iA_j}} - L_{Q_{C_KA_j}}) \tag{12}$$

according to Yu et al. [Yu et al. 1999].

The next part is to model a function, called albedo map, on each surface. Yu et al. present the diffuse albedo for a point x on a surface as

$$\rho_d(x) = \frac{\pi D(x)}{I(x)}, \tag{13}$$

where $D(x)$ is the diffuse radiance map and $I(x)$ is the irradiance map.
$D(x)$ gets computed by substracting the specular part from the whole radiance $L(x)$, which we obtained from the photographs. $I(x)$ can be calculated from the direct illumination using the form factors. To minimize errors that can occur due to wrongly estimated high specular components they eliminate the highest observed specular part at different viewing angles.

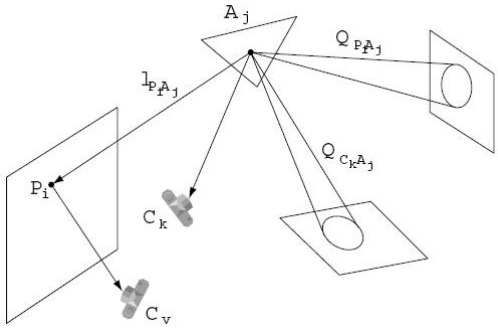

Figure 6: Ray tracing through the patches to find specular reflections. Picture from [Yu et al. 1999]

3.1 Problems and Performance

The running time of their algorithm was about six hours on a 300 MHz PC using 40 HDR images. Whereas I don't think that's possible to do in real-time now, it still shows an interesting approach to get BRDF parameters for a known lighting model.
Also there occured errors in the estimation of the BRDF which affected the diffuse part of the illumination the most.
Another disadvantage is, that a 3D geometric representation of the scene is needed as input parameter.

Figure 7: Result after the first iteration. All specular components are zero.

Figure 8: Result after all iterations. Both pictures from [Yu et al. 1999]

4 Image-Based Rendering of Diffuse, Specular and Glossy Surfaces from a Single Image

This approach, developed by Boivin and Gagalowicz, addresses the problem in an iterative way [Boivin and Gagalowicz 2001]. Every iteration step adds another level of detail to the rendered scene until the result - rendered with Ward's BRDF reflectance model [Ward 1992] - looks well enough.
The input for this algorithm are a simple photograph and a 3D geometric model of the scene. Note that no HDR picture is needed. The 3D geometrical model is built from the single photograph which they achieved with Alias Wavefront's Maya modeler and which took them six hours. The 3D geometrical model of the scene includes camera position and lightsources which are not exact but approximated. Another idea is to add the lightsources manually to the model. For the Ward reflectance model (which was also used in the paper I presented in Section 3) five parameters are needed: the diffuse reflectance ρ_d, the specular reflectance ρ_s, the anisotropy direction $\vec{x}$ and the anisotropic roughness parameters α_x and α_y [Boivin and Gagalowicz 2001].

As stated before the algorithm works iteratively and each iteration step is refined several times. Each subsequent state refines the previous one based on an error picture between the (offscreen) rendered picture and the original photograph. The algorithm runs through a couple of assumptions where the next step is only computed if the error value from the previous picture is over a certain threshold. The assumptions are that the surface is

1. perfect diffuse
2. perfect specular
3. non-perfect specular
4. diffuse and non-perfect specular
5. isotropic (rough)
6. anisotropic (rough)
7. textured

Only if the assumption before is proven wrong with the error picture the next assumption is being taken and a new picture gets rendered and compared afterwards.

4.1 Perfect diffuse

The error ε is computed as the ratio between the average of the radiances from an object (more exactly its a group of objects) in the real picture and in the synthetic one.

$$\widehat{\varepsilon_j} = \frac{\widehat{B_{o_j}}}{\widehat{B_{n_j}}} = \frac{\widehat{T^{-1}(P_{o_j})}}{\widehat{T^{-1}(P_{n_j})}} \tag{14}$$

where $\widehat{B_{o_j}}$ and $\widehat{P_{o_j}}$ are the averages of the radiances and the pixels of an object j in the real image and the index n marks the averages of the same object j in the synthetic image. $T()$ is a γ correction camera transfer function.
The diffuse reflectance ρ_d can be corrected iteratively now with help of this error value. For each rerendering iteration k:

$$\begin{aligned} \rho_{di_{k+1}} &= \rho_{di_k} \times \widehat{\varepsilon_i} \\ \rho_{di_{k+1}} &= \rho_{di_k} \times \frac{\sum_{j=1}^{n_i} f(\widehat{\varepsilon_j}) \cdot (\widehat{\varepsilon_j} \times m_j)}{\underbrace{\sum_{j=1}^{n_i} f(\widehat{\varepsilon_j}) \cdot m_j}_{\neq 0}} \end{aligned} \tag{15}$$

where

$$f(\widehat{\varepsilon_j}) = \begin{cases} 0, & \text{if} \quad \widehat{\varepsilon_j} \geq (1+\lambda) \cdot md \\ 1, & \text{else} \end{cases}$$

and is used to eliminate problems caused by smaller objects which are more affected by noise. $\widehat{\varepsilon_i}$ and $\widehat{\varepsilon_j}$ are the total error values between the original and the synthetic image for group i and object j, n_i is the number of objects for group i, md is the median of the erros, λ is the authorized dispersion criteria and finally m_j is the number of pixels covered by the projection of object j [Boivin and Gagalowicz 2001].
The steady refinement of ρ_d can be seen in Figure 9.

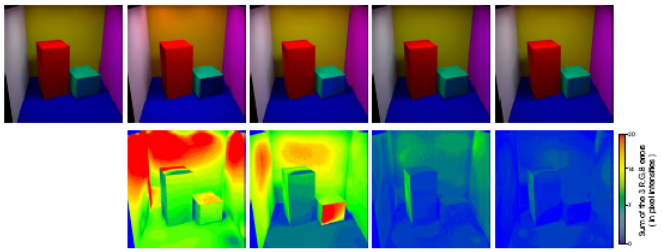

Figure 9: In the top row are from left to right the refined images based on the error pictures below. Picture from [Boivin and Gagalowicz 2001].

If the perfectly diffuse assumption failes (i.e. the error is too high) the second assumption is tried (perfectly specular = all objects are mirrors). This is easy to accomplish as you simply have to set the diffuse reflectance $\rho_d = 0$ and the specular reflectance $\rho_s = 1$ and replace all the ρ_d in Equation 15 with ρ_s.

4.2 Diffuse and specular

If there are still objects that have a high error value, it is now considered to be diffuse and specular (but no roughness is so far assumed) although those surfaces do not appear so often. To get good approximations to these surface reflectances the error value has to be minimized.

$$(T^{-1}(I_{synth}) - T^{-1}(I_o))^2 = \sum_{i=1}^{nbg} (\rho_d \cdot B_d + \rho_s \cdot B_s - T^{-1}(I_o))^2$$

where *nbg* is the number of pixels covered by the group projection. Now the minimization

$$\begin{pmatrix} \rho_d \\ \rho_s \end{pmatrix} = \begin{pmatrix} \sum_{nbg} B_d T^{-1}(I_o) \\ \sum_{nbg} B_s T^{-1}(I_o) \end{pmatrix} \begin{pmatrix} \sum_{nbg} B_d^2 & \sum_{nbg} B_d B_s \\ \sum_{nbg} B_d B_s & \sum_{nbg} B_s^2 \end{pmatrix}^{-1}$$

has a solution for each wavelength R, G, B [Boivin and Gagalowicz 2001].

4.3 Isotropic surfaces

Now the surfaces are assumed to have a certain roughness factor α. This roughness factor is considered in Ward's BRDF Model [Ward 1992]. Now ρ_d, ρ_s and α have to be minimized. Boivin and Gagalowicz did this with the downhill simplex method [Nelder and Mead 1965]. As the accuracy does not have to be that high (10^{-2} for ρ_d and ρ_s and 10^{-4} for α respectively) the parameters can be found within two minutes [Boivin and Gagalowicz 2001].

4.4 Anisotropic surfaces

Now all five parameters of Ward's BRDF model have to be taken into account: The diffuse and specular reflectances ρ_d and ρ_s, the anisotropy direction $\vec{x}$ and the roughness factors α_x and α_y. As they already computed the rather independant ρ_d and ρ_s, they do not have to calculate them anew.
Minimization of the remaining three parameters does not work and only makes the error bigger as no global minimum can be found by the downhill simplex method used in the isotropic case. So they determined $\vec{x}$ from the original picture. Their algorithm to that works as follows:

1. Assume that the anisotropic surface is a perfect mirror
2. Estimate difference between the original image and the synthetic one
3. In this difference picture find the part of the mirror where the specular reflection is "extended".
4. Compute an index buffer for this mirror of all objects visible from it
5. Find a reference surface which covers the largest area of the mirror while being closest to it in a manner that the ratio Area(reflected surface)$/d(S,p)$ is maximized
6. Sample the anisotropy direction creating a number of vectors, where each one determines a direction to traverse the error image and compute the average of the standard error derivations computed in the error image, around the normal to the anisotropic surface
7. Select the direction for which the average error becomes smallest
8. This direction equals $\vec{x}$

where $d(S,p)$ is the Euclidian distance between the center of gravity of the selected surface and the center of gravity of the mirror [Boivin and Gagalowicz 2001].
Now that their algorithm computed the anisotropy direction the remaining two parameters can be determined via the previously mentioned downhill simplex method.

4.5 Textured surfaces

This is their final assumption if all previous assumptions failed to provide low error values. As the textures of the objects in the real image are already lit by the lightsources they can not simply take those textures as they would be overlit by the lightsources that are already present in the virtual image. So they introduce the notion *radiosity texture* that balances the extracted texture with an intermediate texture in order to minimize the error between the real and the synthetic image, where these error minimizations follow the principle I have shown in Section 4.1 (the iterations to minimize the diffuse reflecatance error) [Boivin and Gagalowicz 2001].
To increase the speed of this method they propose several alterations of the algorithm. For example, if the error after the specular assumption is higher than 50% the algorithm goes directly to the textured case as the isotropic and anisotropic cases last the longest (almost 4 hours for the anisotropic case).

Figure 10: Result of the algorithm presented by Boivin and Gagalowicz. These images show the usability of their approach for augmented reality applications as there have been made some changes from the original image. The image on the upper left side shows the original image with some removed objects. The other ones show the same scene under novel lighting conditions, new viewpoints or with new objects. Picture from [Boivin and Gagalowicz 2001].

4.6 Performance and Problems

The recovery of the image in the anisotropic case took more than four hours. With the enhancments they made the algorithm does not have to try the isotropic and anisotropic cases as they do not promise to deliver better results. So the two most time consuming parts of their algorithm can be skipped if the error after the specular assumption is too high.
The rendering of Figure 10 took about half an hour. The preprocessing was of course more time consuming, the inverse algorithm took them 4 hours and 40 minutes, where 4 hours alone have been

spent to recover the anisotropy parameters for the aluminium surface [Boivin and Gagalowicz 2001].

5 Photorealistic rendering for augmented reality using environment illumination

Agusanto et al. used an image based lighting approach [Agusanto et al. 2003], [Debevec and Malik 1997]. They acquire the scene radiance with HDR photography and process the data afterwards in their rendering framework and render the pictures with the Phong illumination model [Phong 1975].
Image based lighting means that radiance maps taken from a real (HDR) picture are used to get the scene illumination. Those radiance maps can be used to correctly illuminate virtual objects rendered into the scene.
Other than Ritschel and Grosch who used two HDR cameras, Agusanto et al. used photography and a light probe. The light probe is a calibrated reflectance sphere (a mirror ball) made of chrome. A big advantage is the low cost of the utilities. The ball is placed in the scene where the virtual objects will be positioned later and pictures (with varying position, shuttertimes and exposure values, see Figure 11) are taken. They made low dynamic range pictures and assambled them with HDRShop [HDRShop] developed by Debevec et al. into one single HDR picture.
Of course the lighting conditions of the AR application needs to be the same it was during the recording of the scene with the light probe. Plus the camera should ideally be aligned to the viewer's point of view.
As they also obtain the interreflections between all objects in the scene, these radiance maps can be used as environment illumination maps in hardware based rendering which is employed for AR.

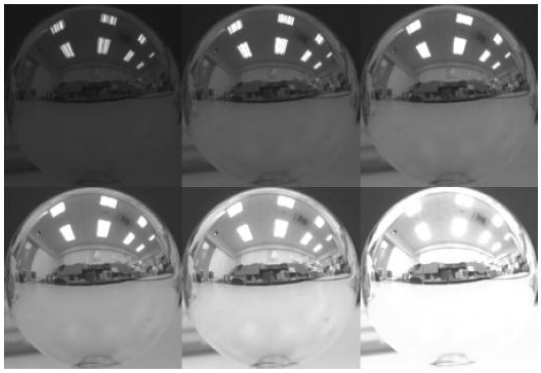

Figure 11: The lightprobes photographed at different shutter times. Picture from [Agustanto et al. 2008]

5.1 Creating illumination maps

To keep flexibility within their approach the environment maps are derived from the radiance maps. Otherwise they would have to use a sphere with different surfaces and materials for different objects. Instead these different types of spheres (diffuse, specular, glossy...) are simulated in a virtual environment which is basically just a box textured with the previously obtained illumination map and the demanded type of sphere in the middle of it (in a very small ratio from 1:50 to 1:500).
The viewing vector needs to be directed at the center of the sphere. The surrounding box serves as area light source for the global illumination algorithm (ray tracing) to obtain the synthetic illumination map.

5.2 Environment map prefiltering

To obtain the diffuse and the specular illuminations the environment maps run through a prefiltering process. This process is the solving of the rendering equation (Equation 2). Agusanto et al. used an approach presented by Kautz et al. [Kautz et al. 2000].
A prefiltered environment map captures all the reflected radiance to a certain direction.

$$L_{glossy}(x;\vec{v},\vec{n},\vec{t}) = \int_{\Omega} f_r(\vec{\omega}(\vec{v},\vec{n},\vec{t}),\vec{\omega}(\vec{l},\vec{n},\vec{t}))L_i(x;\vec{l}) < \vec{n},\vec{l} > d\vec{l}, \tag{16}$$

where $\vec{v}$ is the viewing direction and $\vec{l}$ is the light direction, $\{\vec{n},\vec{t},\vec{n}\times\vec{t}\}$ is the local coordinate frame of the reflective surface, $\vec{\omega}(\vec{v},\vec{n},\vec{t})$ represents the viewing direction and $\vec{\omega}(\vec{l},\vec{n},\vec{t})$ the light direction relative to that frame. The pre-filtered environment map now stores the radiance of light reflected towards the viewing direction $\vec{v}$ which is computed by weighting the incoming light L_i (which is basically the unfiltered environment map) from all direction $\vec{l}$ with the BRDF f_r [Kautz et al. 2000].

5.2.1 Diffuse Environment Maps

Miller proposed to use only purely diffuse BRDF to prefilter environment maps [Miller and Hoffman 1984]. So from Equation 16 they get

$$L_{diffuse}(x;\vec{v},\vec{n},\vec{t}) = \int_{\Omega} k_d L_i(x;\vec{l}) < \vec{n},\vec{l} > d\vec{l}. \tag{17}$$

Now all dependencies except for the normal vector are dropped and the resulting two dimensional environment map is:

$$L_{diffuse}(x;\vec{n}) = k_d \int_{\Omega} L_i(x;\vec{l}) < \vec{n},\vec{l} > d\vec{l}. \tag{18}$$

and stores all diffuse illumination at point x [Kautz et al. 2000].

Agusanto et al. used several prefiltering techniques. Apart from the diffuse map they also applied the Phong environment map since they used the Phong illumination model for their rendering.

5.2.2 Phong Environment Map

The environment map for Phong's BRDF model can be expressed as:

$$\begin{aligned} L_{phong}(x;\vec{v},\vec{n},\vec{r}) &= \int_\Omega k_s \frac{<\vec{r}_v(\vec{n}),\vec{l}>^N}{<\vec{n}\vec{l}>} L_i(x;\vec{l}) < \vec{n},\vec{l} > d\vec{l} \\ &= k_s \int_\Omega < \vec{r}_v(\vec{n}),\vec{l} >^N L_i(x;\vec{l}) d\vec{l}. \end{aligned} \quad (19)$$

For the usage without prefiltering (as is supported in OpenGL) the indexing can be done just with the reflection vector $\vec{r}_v$:

$$L_{phong}(x;\vec{r}_v) = k_s \int_\Omega < \vec{r}_v,\vec{l} >^N L_i(x;\vec{l}) d\vec{l}. \quad (20)$$

For the complete illumination model which Agusanto et al. used, they follow a method from Miller [Miller and Hoffman 1984] and Heidrich [Heidrich and Seidel 1999] and use a weighted sum of a diffuse and a Phong environment map with a Fresnel term that varies the ratio between the diffuse reflection and the lobe of the specular reflection so that a wider variety of materials can be rendered [Kautz et al. 2000]. The final environment map is:

$$L_o(\vec{r}_v,\vec{n}) = F_d(< \vec{r}_v,\vec{n} >) L_{diffuse} + F_p(< \vec{r}_v,\vec{n} >) L_{phong} \quad (21)$$

5.3 Rendering

Rendering is done within a multi pass rendering framework to render different effects like antialiasing, stencil buffer and so on. The framework presents itself as follows:

for a number of image samples **do**
  Jitter the camera
  Enable stenceling / blending operation
  for a number of passes **do**
    Add effects to the polygon model
  end for
  Render the polygon model
  Do stencil / blending test
  Perform accumulation buffer operation
end for

Some operations - like sampling of images - are only needed if photorealism is required as they reduce the speed remarkably. The number of images being sampled can be reduced to one if photorealism is not that important.
As this framework enables stencil buffer (e.g. for shadow maps) we don't need global illumination to obtain shadows.

Now their implementation (with OpenGL commands) of this multipass framework is presented. The most important commands of their implementation are:

1. Enable blending
2. Transform the object
3. Enable 2D texture mapping
4. Set blending functions *(GL_ONE, GL_ZERO)*
5. Render the virtual object
6. Enable environment mapping
7. Set environment mapping to *GL_DECAL* mode
8. Generate the sphere mapping texture coordinates
9. Set blending function
   (GL_DST_COLOR, GL_ONE_MINUS_SRC_ALPHA)
10. Render the virtual object

and do this for all virtual objects.

The reason behind the multiple passes is to add one photorealistic effect after the other in each pass. In the second pass for example, the previously obtained illumination map is used to illuminate the virtual object. With the use of the derived environment map they don't need to specify light sources for the rendering process. For shadowing the lightsources are being approximated by the also previously taken light probe image.

The algorithm has been tested using ARToolKit [Shared-Space-ARToolKit.] and 3D polygon models (The programming language was C with OpenGL).

5.4 Performance and Problems

All models that have been rendered with this algorithm could be drawn at almost real-time (> 17 fps). As this is a seven year old paper (using just a 400 MHz machine) this surely will be much faster today.

The main problem is that a lot of computations have to be done before the actual rendering is done. The biggest advantage of this approach is the high photorealism it provides (see Figure 12)

Figure 12: Diffuse and glossy models rendered at 17 fps. Picture from [Agusanto et al. 2008]

6 Recovery of material under complex illumination conditions

Wu et al. presented a method on how to recover the BRDF for RADIANCE's low parameter reflectance model, which also uses Ward's model, for a real homogenous object under complex lighting conditions from a HDR photograph of the object and one of the environment to find the lightsources [Wu et al. 2004].
Important is that the object is lit in a way that enough specular highlights are seen, so that only one HDR image is sufficient. The illumination field is computed via environment maps taken from different viewpoints.
Again their aim is to recover f_r (here ρ) - the BRDF - from Equation 2, where all other variables are known. The parameters they

need for their modified RADIANCE reflectance model are specular, diffuse and directional diffuse refleactance and transmission. The BRDF model they used can be expressed as

$$f_r = \max(0,\vec{q}\cdot\vec{n}_p)\left[\frac{\rho_d}{\pi}+\rho_s\right]+\max(0,-\vec{q}\cdot\vec{n}_p)\left[\frac{\tau_d}{\pi}+\tau_s\right] \quad (22)$$

where

$$\begin{aligned}
\rho_d &= pC(1-a_4)\\
\rho_s &= r_s\frac{f_s(\vec{q})}{\sqrt{(\vec{q}\cdot\vec{n}_p)(-\vec{v}\cdot\vec{n}_p)}}\\
\tau_d &= a_6(1-r_s)(1-a_7)pC\\
\tau_s &= a_6a_7(1-r_s)\frac{g_s(\vec{q})}{\sqrt{(-\vec{q}\cdot\vec{n}_p)(-\vec{v}\cdot\vec{n}_p)}}\\
r_s &= \begin{cases} a_4 & \text{plastic}\\ \{a_1a_4,a_2a_4,a_3a_4\} & \text{metal}\end{cases}
\end{aligned}$$

where $\vec{q}$ is the direction from the surface point to a light source sample, $\vec{v}$ is the direction of an eye ray, $\vec{n}_p$ is the surface normal at the point p, C is the surface colour, p is the material pattern and $a_i(i=1,\ldots,7)$ are parameters. The ρ-parameters define the reflection, the τ-parameters the transmission coefficients [Wu et al. 2004].
The two functions $f_s(\vec{q})$ and $g_s(\vec{q})$ have the form

$$f_s(\vec{q}) = \begin{cases} \frac{e^{[(\vec{h}\cdot\vec{n}_p)^2-\|\vec{h}\|^2]/(\vec{h}\cdot\vec{n}_p)^2/\alpha}}{4\pi\alpha}, & \text{isotropic}\\ \frac{1}{4\pi\sqrt{\alpha_x\alpha_y}}exp\left[\frac{\frac{(\vec{h}\cdot\vec{x})^2}{\alpha_x}+\frac{(\vec{h}\cdot\vec{y})^2}{\alpha_y}}{(\vec{h}\cdot\vec{n}_p)^2}\right], & \text{anisotropic}\end{cases} \quad (23)$$

and as they only consider opaque materials for the anisotropic case, for the isotropic case

$$g_s(\vec{q}) = \frac{e^{(2\vec{q}\cdot\vec{t}-2)/\beta}}{\pi\beta}, \quad (24)$$

where

$$\begin{aligned}
\alpha_x = \alpha &= a_5^2+\frac{\omega}{4\pi}\\
\alpha_y &= a_6^2+\frac{\omega}{4\pi}\\
\beta &= a_5^2-\frac{\omega}{\pi}\\
\vec{t} &= \frac{\vec{v}}{\|\vec{v}\|}
\end{aligned}$$

and $\vec{h}$ is the bisection vector between the incident light ray and the eye ray [Wu et al. 2004].

After the acquisition of the illumination maps they now proceed to recover the wanted materials for the object which is lit by known lighting that is represented as an illumination field. Similar to the approach described in Section 4 the recovery of the material parameters is done via the minimization of a difference value between the real (I_r) and a synthetic rendered (I_o) image of the object. The difference in the mean of least squares is

$$\begin{aligned}
\chi^2 &= \|I_o-I_r\|^2=\textstyle\sum_i\|p_{Oi}-p_{ri}\|^2=\\
&= \textstyle\sum_i[(r_{oi}-r_{ri})^2+(g_{oi}-g_{ri})^2+(b_{oi}-b_{ri})^2]
\end{aligned} \quad (25)$$

where r, g and b mark the colour values of a certain point p. This is a nonlinear optimization problem that Wu et al. solved with simulated annealing algorithm. The algorithm works with a set of initial parameter values to optimize them and to reduce χ^2 step by step until a global minimum is found. These calculated parameters are then used to render the object with ray tracing. A result of their work can be seen in Figures 13 and 14.

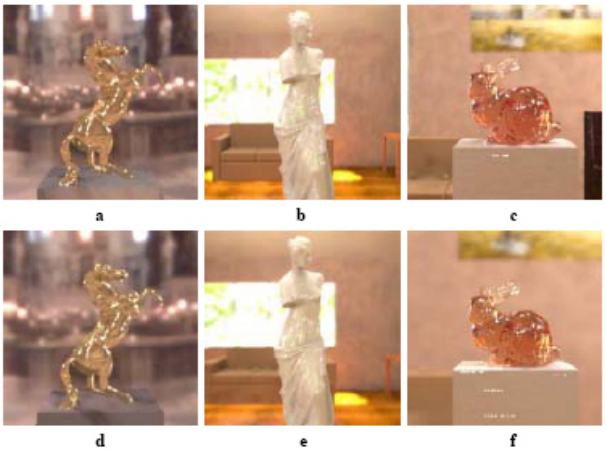

Figure 13: a), b) and c) show the real objects, d), e), and f) the rendered ones after the last optimization step. Picture by [Wu et al. 2004]

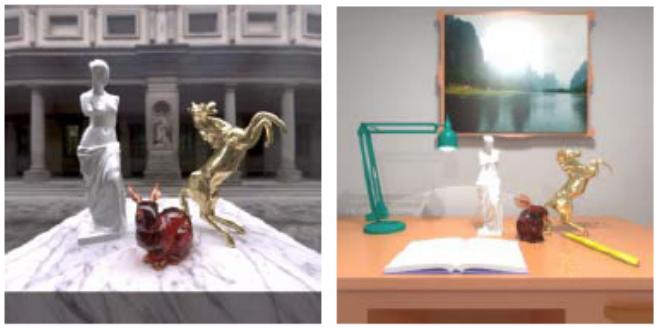

Figure 14: Left: Virtual objects rendered into a real (augmented reality) scene. Right: Virtual objects rendered into a virtual scene. Picture by [Wu et al. 2004]

6.1 Performance and problems

The recovery of the materials took them about 2 to 3 hours on a Dell Dimension 4100 with a 667 MHz CPU and with 128 MB of working memory [Wu et al. 2004]. The method is similar to the method by Boivin and Gagalowicz [Boivin and Gagalowicz 2001] and is not remarkably faster. On the other hand they do not need a 3D geometrical model of the scene which makes the precomputations cheaper. The illumination calculations (extraction of lightsources for the illumination map to light the virtual objects) they presented

were also rather interesting and take some computation cost but they are not topic of this thesis.
As they also noted they can not reconstruct all materials especially not high frequent materials as they would require additional light-sources in the real scene.

7 A Framework for Automatically Recovering Object Shape, Reflectance and Light Sources from Calibrated Images

Mercier et al. [Mercier et al. 2007] present a method for recovering object shapes, reflectance properties (for the modified Phong model [Lewis 1994]) and the position of light sources from a set of images. In my thesis only the surface and reflectance recovery is going to be discussed. The modified Phong model is expressed as

$$L_r = \frac{L_s k_d}{\pi r^2} \cos\theta + \frac{(n+2) L_s k_s}{2\pi r^2} \cos\theta \cos^n \phi \tag{26}$$

where L_r is the radiance reflected, L_s is the radiance emitted by a light source S arriving at P, r is the distance between S and P, θ is the angle between the surface normal and the direction of the light source, ϕ is the angle between the mirror reflection direction and the actual reflection direction and again k_s and k_d are the specular and diffuse parameters and n is the specular exponent.

For each of the images in the image-set the position and orientation of the camera have to be known (see Figure 15).

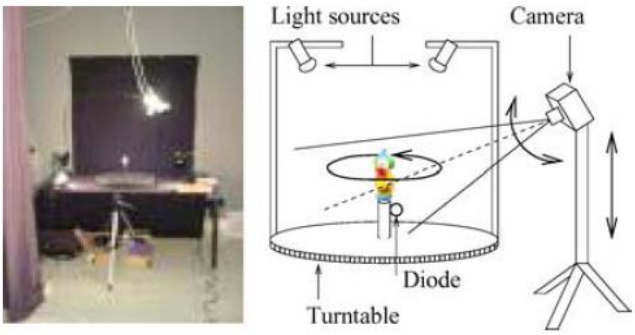

Figure 15: Camera setup with fixed light sources. [Mercier et al. 2007]

They made different images for different purposes. In Figure 15 you see that they made an overexposed image for segmentation of the surfaces and a second image for reflectance and for the estimation of the lightsources.

Their first step is to acquire the object shape from these images using a shape to silhouette approach. With the help of image pixels they used the marching cubes algorithm for providing the polygonal surface and the surface normals (that need to be estimated for each voxel) that are later needed to estimate the BRDF parameters.
After they have acquired the polygonal mesh of the object they now proceed to the estimation of the light source directions which can be found using specular highlights seen on images. They also note that diffuse or specular self-interreflections sometimes disturb the BRDF estimation [Mercier et al. 2007].

Figure 16: a) The overexposed image, b) normal image, [Mercier et al. 2007]

During the estimation of the lightsources they also present an identification algorithm to provide the BRDF coefficients again using an error function E_a.
They split the estimation of lightsources into two classes. The first one applies to diffuse the second to the glossy surfaces. To find the type of surface a variation coefficient V^{class} is computed from the radiance samples:

$$V^{class} = \sqrt{\frac{\sum_{i=1}^{NbV} \sum_{i=1}^{NbL_i} \left(\frac{L_{i,j} - L_i^{moy}}{L_i^{moy}} \right)}{\sum_{i=1}^{NbV} NbL_i}} \tag{27}$$

where NbV represents the number of voxels in the class, V_i is the ith voxel, $L_{i,j}$ is the jth radiance sample of V_i, and L_i^{moy} is the average radiance of V_i; NbL_i corresponds to the number of radiance samples in the voxel V_i [Mercier et al. 2007]. Now for a perfectly diffuse surface Equation 27 equals zero. V^{class} increases directly proportional to the specular aspect of the surface i.e. that it is higher the glossier a surface is. Some examples for V^{class}:

- $V^{class} < 0.15 \Rightarrow$ perfectly diffuse
- $V^{class} > 0.30 \Rightarrow$ glossy
- $0.15 \leq V^{class} \leq 0.30 \Rightarrow$ no decision possible. For this class Mercier et al. apply algorithms for both diffuse and glossy surfaces.

For both surface types a point light and a directional light source are estimated according to the error function E_a [Mercier et al. 2007]:

$$\sum_{i=1}^{NbV} \sum_{j=1}^{NbL_i} \left[\left(\frac{L_s k}{\pi r^2} \cos\theta_i + \frac{(n+2) L_s k_s}{2\pi r^2} \cos\theta_i \cos^n \phi_{i,j} \right) - L_{i,j} \right]^2 \tag{28}$$

where $L_{i,j}$ corresponds to the radiance sample j of the voxel V_i. The reflectance paramters $L_s k_d$, $L_s k_s$ and n are not known and need to be estimated. Mercier et al. applied an identification algorithm with the help of a gradient descent method in order to minimize E_a and hereby find the BRDF parameters. The parameters are chosen this way that E_a becomes as low as possible and the type of surface is known. This is applied for each class and to refine the BRDF parameters a final identification algorithm is applied again using the error E_a. To reduce noise and grazing angles, Mercier et al. used only radiance samples corresponding to a small incidence angle lower than 45 degrees (see Figure 17).

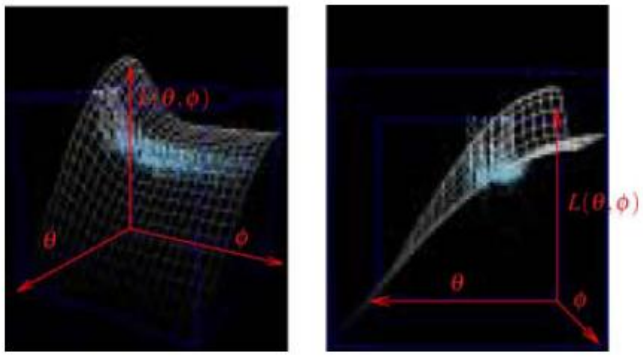

Figure 17: Estimated BRDF together with actual radiance samples [Mercier et al. 2007]

7.1 Performance and Problems

They used a Dual Intel Xeon 2.4 GHz processor with 1 GB of working memory. The BRDF estimation of the clown Figure (see Figure 18) took 6 minutes and 30 seconds precomputation time.

Figure 18: Result of the algorithm with sharp specular highlights [Mercier et al. 2007]

The approach by Mercier et al. has also certain limitations. For example is it hard to acquire the surface proberties if the object posseses a variety of textures. Also diffuse interreflections are ignored hence certain artifacts can affect the shading [Mercier et al. 2007].

8 Recovering surface reflectance and multiple light locations and intensities from image data

Xu and Wallace presented a method to recover reflectence properties from multiple objects using a 3D image. Their approach provides the diffuse and constant specular reflectance parameters from object images [Xu and Wallace 2008]. They also recover the light source parameters.

To get the surface geometry they used an active 3D scanner and a stereo pair of CCD cameras. They split their approach into two steps. The first step is to get the light source parameters and the specular reflection for the Phong-Blinn reflection model with the simplified formula for the specular irradiance which also considers the light intensity (see Equation 5 for the original formula)

$$I_{spec} = k_s L(P)(H \cdot N)^n \tag{29}$$

where $L(P)$ is the light intensity at point P, N is the normal at that point, n is the specular exponent and H is the halfway vector which calculates as

$$H = \frac{l+V}{|l+V|},$$

where l is the normalized vector pointing to the lightsource and V is the viewing vector [Xu and Wallace 2008], [Hearn and Baker 2004].

8.1 Obtaining the specular reflectance

Remember that they used two cameras. They assume that both cameras have the same radiometric constant α_0 and give the image brightness (diffuse and specular reflectance) as

$$I(p) = \frac{\alpha_0 k_d L_0}{|l-P|^3}[(l-P)\cdot N] + \frac{\alpha_0 k_s L_0}{|l-P|^2}(H \cdot N)^n \tag{30}$$

which gives two equations one for each camera where only the halfway vector H differs.

$$\begin{cases} I(p_1) = \frac{\alpha_0 k_d L_0}{|l-P|^3}[(l-P)\cdot N] + \frac{\alpha_0 k_s L_0}{|l-P|^2}(H_1 \cdot N)^n \\ I(p_2) = \frac{\alpha_0 k_d L_0}{|l-P|^3}[(l-P)\cdot N] + \frac{\alpha_0 k_s L_0}{|l-P|^2}(H_2 \cdot N)^n \end{cases} \tag{31}$$

Now for convenience $S_s = \alpha_0 k_s L_0$ and the difference from both camera images is

$$\Delta I(P) = \frac{S_s}{|l-P|^2}[(H_1 \cdot N)^n - (H_2 \cdot N)^n]. \tag{32}$$

The part for the diffuse reflectance disappeared as the diffuse reflectance is the same for each viewpoint and only the specular reflectance differs. Now Xu and Wallace compute the difference ε_r between the meassured and the predicted values

$$\begin{aligned} \varepsilon_r(P_i) &= \Delta I_m(P_i) - \Delta I(P_i) \\ &= \Delta I_m(P_i) - \tfrac{S_s}{|l-P_i|^2}[(H_{i1} \cdot N_i)^c - (H_{i2} \cdot N_i)^n] \end{aligned} \tag{33}$$

where $i = 1, \ldots, n$, n indicates the number of successfully measured brightness values in the paired images [Xu and Wallace 2008]. Now this error gets again minimized. The unkown variables are the X, Y, Z values of l, the specular coefficient S_s and the specular exponent c. The minimization function is called $\mathbf{f}$.

$$\mathbf{f}_{lr} = \sum_{i=1}^{n} \left[\Delta I_m(P_i) - \sum_{j=1}^{m} \left(\frac{S_{sj}}{|l_j - P_i|^2}[(H_{ij1} \cdot N_i)^c - (H_{ij2} \cdot N_i)^c] \right) \right]^2. \tag{34}$$

To solve this minimization Xu and Wallace used a gradient descent least-square optimization procedure on **f** using the Levenberg-Marquardt method. To compute the initial values for $S_s j$ (the j parameter is for multiple lightsources) the following steps have to be made. First a linear system is obtained by approximating the measured image brightness difference ΔI_m using the predicted values ΔI.

$$\begin{bmatrix} \frac{1}{|l-P_1|^2}[(H_{11}\cdot N_1)^c-(H_{12}\cdot N_1)^n] \\ \vdots \\ \frac{1}{|l-P_n|^2}[(H_{n1}\cdot N_n)^c-(H_{n2}\cdot N_n)^n] \end{bmatrix}[S_s]=\begin{bmatrix} \Delta I_m(P_1) \\ \vdots \\ \Delta I_m(P_n) \end{bmatrix} \quad (35)$$

This equation has the form $B_{lr}S_s = \Delta I_m$. The least-squares solution for this linear system is determined from [Xu and Wallace 2008]

$$S_s = (B_{lr}^T B_{lr})^{-1} B_{lr}^T \Delta I_m. \quad (36)$$

8.2 Obtaining the diffuse reflectance

Following the method to gain the specular reflectance Xu and Wallace now use Equation 30 to estimate the diffuse parameter.

$$\begin{cases} \delta_1(P_i)I_m(p_{i1}) = \delta_0(P_i)\delta_1(P_i)\left[\frac{S_{di}S_s}{|l-P_i|^3}[(l-P_i)\cdot N_i]+\frac{S_s}{|l-P_i|^2}(H_{i1}\cdot N_i)^n\right] \\ \delta_2(P_i)I_m(p_{i2}) = \delta_0(P_i)\delta_2(P_i)\left[\frac{S_{di}S_s}{|l-P_i|^3}[(l-P_i)\cdot N_i]+\frac{S_s}{|l-P_i|^2}(H_{i2}\cdot N_i)^n\right] \end{cases} \quad (37)$$

By adding those two equations $S_{di}, (i = 1,\ldots,n')$ are estimated by

$$S_{di} = \frac{[\delta_1(P_i)I_m(p_{i1})+\delta_2(P_i)I_m(p_{i2})]}{\delta_0(P_i)(\delta_i(P-I)+\delta_2(P_i))S_s[(l-P_i)\cdot N_i]} - \frac{\delta_0(P_i)S_s|l-P_i|[\delta_1(P_i)(H_{i1}\cdot N_i)^n+\delta_2(P_i)(H_{i2}\cdot N_i)^n]}{\delta_0(P_i)(\delta_i(P-I)+\delta_2(P_i))S_s[(l-P_i)\cdot N_i]} \quad (38)$$

where n' is the number of successfully measured brightness values, the $\delta_x(P_i)$ denote visibiltiy of the point P_i to the light (δ_0), the first (δ_1) and the second image (δ_2) [Xu and Wallace 2008].

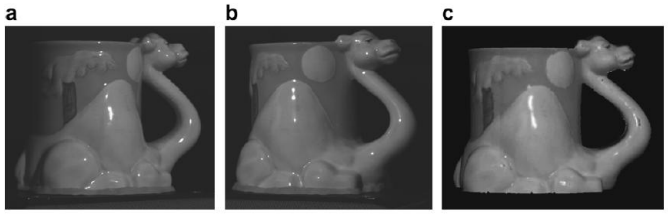

Figure 19: a) and b) are the two stereo images of the camel cup, c) is the rendered image with the estimated parameters from a new viewpoint [Xu and Wallace 2008]

8.3 Performance and Problems

A result of Xu and Wallace's work can be seen in Figure 19. The difference between the estimated specular parameters and the ground truth was (at 1% additive noise) 4.76% for S_s and 1.1% for c. This image had only one point light source. The error goes up as the number of lightsources increases.

9 Online Estimation of Diffuse Materials

Ritschel and Grosch presented a way to get diffuse parts of BRDFs at runtime from digital photographs [Ritschel and Grosch 2008]. They do not really work in real-time environments but expect their approach to also work in real-time. The only difference really is the parametrization of the model (which has to be done automatically at real-time). They used two HDR video cameras to get the diffuse materials. The equation for the outgoing radiance on a photograph at a surface point is

$$L_0 = \int_{2\pi} f_r(\omega_i,\omega_0)L_i(\omega_i)cos(\theta_i)d\omega_i \quad (39)$$

where f_r is the BRDF of the surface, L_i is the incoming radiance from direction ω_i and $cos(\theta_i)$ is the cosine of the angle between incoming direction and surface normal.

If we follow the calculations of Debevec [Debevec 1998] the initial values for L_0 are being approximated as perfectly diffuse ($f_r = 1$). So after putting f_r in front of the integral and dividing by the incoming radiance L (and inverting the whole equation afterwards) we get f_r for each part of the image as a result of the fraction

$$f_r = \frac{L}{\int_{2\pi} L_i(\omega_i)cos(\theta_i)d\omega_i}. \quad (40)$$

The previously mentioned HDR cameras are positioned as follows. One is observing the object whose BRDF shall be approximated and the second is filming the light source (see Figure 20). This *light-camera* is at a fixed position and records the whole environment illumination with a fisheye lens. The *object-camera* should be moved as close to the object as possible so we get the same illumination on the virtual object as on the camera.

The *object camera* should take pictures from different viewing angles and capture the illumination of the objects it observes. The rotation of the objects needs to be known. Also a marker is placed besides the object to track camera position and orientation. This happens via optical tracking with ARToolkit [Shared-Space-ARToolKit.]. They used a couple of markes with known position to increase the precision of the process.
But first the synthetic object is required to be in a polygonal mesh representation. As said before, Ritschel and Grosch did this offline (manually), but this can be done using an automatic parametrization as presented by Levy [Lévy et al. 2002].

The images captured with this camera run through a couple of processes discussed in the following. The software side of their procedure is divided into two steps: *Inverse Texturing* (storing the camera images to a texture) and *Inverse Lighting* (processing these textures to one final reflectance texture).
As their approach only works for diffuse lighting, they propose to factorize the software steps into orthogonal components to get the specular part as well.

Figure 20: Camera setup by [Ritschel et al. 2006]

Inverse Texturing. The pixels sampled are being converted to texels (from vertex- to texturespace) then drawn into a texture. The next step is to discard all pixels before the near-plane of the camera. Then the perspective has to be transformed through linear interpolation and translation. Finally the texture gets drawn onto the object as a triangle. A vertex program transforms the required vertices and exchanges them with the texture coordinates. So the name explains as they sample the pixels (RGB color) from an image to draw into a texture rather than sampling from a texture to draw the triangles onto the image.
Then all the pixels that are not seen from the viewer's angle are being cut out via an *id buffer*. To get this buffer the whole scene gets rendered from the object camera's point of view using an OpenGL command to encode the facets as a certain color with depth buffering and culling enabled. This procedure works better than a depth buffer as it is only required to know whether a texel is equal depth (and not behind another texel as it is required by depth buffering). The next thing to do after the first rendering step is to compare all the facets. If id's of two facets are unmatched the first facet's alpha is set to zero (i.e. it's not going to appear in the final image). They also propose to use this technique to occlude objects that don't belong to the final image (this only works if their geometry is known).

To get a decent alpha channel of the texture a so called *confidence* (i.e. weights) is also stored depending of the viewing angle and distance. The confidence is a sum of

- Viewing distance λ_d (larger value for closer views)
- Viewing angle λ_a (larger value for normal angle)
- Distance to camera image center λ_p (larger values near the center)
- Camera speed λ_v (larger values for slower motions)

The borders of LDR images are darker and blurred, therefore their confidence is lesser than the one in the middle of the object. Also they experienced problems while determining λ_v as motion blurring appears when moving the camera not continously. They call these t observation textures $A_i = (A_1, A_2, ...A_t)$.

Inverse Lighting. From A_t they determine a radiance atlas R_t storing all the diffuse values. To get these values Equation 40 is used (the radiance L is being observed, the irradiance (the bottom part of the fraction) needs to be simulated). When they finally have t textures and radiance maps the need to compute a final map R_{final}. This is done iteratively. First they set

$$R_{final} = R_0.$$

Now a small part of the following R_i are added at every step t.

$$R'_{final} = (1-\lambda)R_{final} + \lambda R_t$$

λ can be chosen arbitrary but $\lambda = 0.02$ gives good results. This whole operation is an averaging of all R_i to determine a final radiance map and eliminating noise.

Illumination can now be determined by using a technique presented by Havran [Havran et al. 2005]. Basically they determine the lightsources recorded by the light-camera (the second camera which is directed at the lightsource as in the previously mentioned model) and use them for environment mapping.

After the determination of the light sources shadows are being drawn via multiple shadow maps (which can happen in real-time). Here the real shadow of the object is being removed and the new computed one is being drawn.

9.1 Performance and problems

Figure 21: A cloned donkey from [Ritschel et al. 2008]

The algorithm presented by Ritschel and Grosch has a performance of 5 fps with a model that has 100 facets and a resolution of 320x240 (results see Figure 21). This can be faster today, as their paper is 2 years old. The most time consuming process was the inverse lighting (70 ms). As this is done on the GPU (on a GeForce 7800) the algorithm can be faster today (e.g. on a GeForce GTX

480). Of course the performance always depends on the number of pixels and texels.
The presented method is not without certain unsteadinesses. The problem of optical tracking (where the error is proportional to the distance from the tracker to the object) is always present.

10 Comparison of the presented methods

The first presented paper by Yu et al. used an inverse global illumination algorithm to approximate the BRDF parameters [Yu et al. 1999]. It is clear that this approach is very expensive and not as practical as more modern approaches which mostly cite Yu et al.'s work. The running time of this work is not that important as their computations can be done during the pre-computation step and we can not say how fast their approach would work with todays GPUs.
The largest part of the papers presented in this thesis used image based methods to estimate the BRDF parameters. Most need an HDR, or couple of LDR pictures of the scene and a 3D model of the scene/of an object. Some papers ([Yu et al. 1999], [Boivin and Gagalowicz 2001], [Agusanto et al. 2003]) need a polygonal mesh as input. The other ones include or reference to a parametrization method which is of course also costly.
So if a 3D polygonal mesh is known those three methods can be applied to approximate or estimate the BRDF. Especially the method from Boivin and Gagalowicz [Boivin and Gagalowicz 2001] provides very good looking results with an algorithm that is not hard to implement.
If the 3D polygonal mesh is not known one of the more modern approaches should be used as some of them ([Mercier et al. 2007], [Wu et al. 2004], [Ritschel and Grosch 2008]) presented a software approach to find the shape of the objects just from images. Xu and Wallace [Xu and Wallace 2008] used an active 3D scanner and a pair of CCD cameras which does not appear to be a low budget solution.
If a fast solution is needed, the approach from Boivin and Gagalowicz [Boivin and Gagalowicz 2001] is the best choice even though a fast approximation provides an image which is not the correct one. If you change the threshold and the error value accordingly you get a rough approximation but after a very short time (if you skip the isotropic and anisotropic assumptions it will always be rather fast).
Some papers ([Agusanto et al. 2003], [Mercier et al. 2007], [Xu and Wallace 2008]) used the Phong illumination model or the Phong-Blinn model respectively. While Phong's model is widely used and simple it is still not physically plausible. Other papers try to make their objects look more realistic and adapt different illumination models. Two papers ([Yu et al. 1999], [Boivin and Gagalowicz 2001]) approximated and estimated parameters for Ward's BRDF model, others ([Wu et al. 2004]) for RADIANCE or a completely different illumination algorithm ([Ritschel and Grosch 2008]).
The realism of the papers was generally rather good. Where some had some problems with artifacts due to optical tracking errors ([Ritschel and Grosch 2008], [Xu and Wallace 2008]) others did not have this problem as they already had the polygonal mesh of the objects, as I have previously mentioned.
The best solution for augmented reality applications was presented by Boivin and Gagalowicz who also presented some ways to deal with their method in augmented reality (setting of a novel viewpoint, changing illumination conditions, adding and removing objects). A very interesting aspect was that isotropic surfaces (a mirror for example) were recognized that way and were not just textured. That means that objects which could be added to the scene would also be reflected in that mirror [Boivin and Gagalowicz 2001].
Most of the other papers dealt only with the recovery of BRDF (or shape) of a certain model as they assumed that the geometry is not known.

11 Conclusion

In my thesis I presented 7 papers that deal with BRDF approximation or estimation. All but Yu et al. [Yu et al. 1999] mention the possibility to use their work in augmented reality applications which was of course the topic and an important point of my thesis.
In each section I summarized the parts of the papers that dealt with BRDF approximation or estimation with respect to their usage in augmented reality. I presented results of the different papers and concluded each section with a short paragraph about the performance (if given in the original work) and problems that occured or might occur when using the corresponding algorithm.
At the end I compared all the relevant papers in respect to input data, running time, the illumination method used (Phong, Phong-Blinn, Ward,...), photorealism, whether or not a rough approximation of an image before the actual computation of the end result is possible, their usage in augmented reality applications and whether the methods are costly or low budget solutions.
All the approaches are rather sophisticated and it might be interesting to find out whether some of the older papers I presented ([Yu et al. 1999], [Boivin and Gagalowicz 2001], [Agusanto et al. 2003]) might work completely on the GPU or generally faster with more modern graphic cards.

References

AGUSANTO, K., LI, L., CHUANGUI, Z., AND NG, W. S. 2003. Photorealistic rendering for augmented reality using environment illumination. In *ISMAR*, IEEE Computer Society, 208–216.

AZUMA, R. 1997. A survey of augmented reality. *Presence 6*, 4, 355–385.

BLINN, J. F., AND NEWELL, M. E. 1976. Texture and reflection in computer generated images. *Communications of the ACM 19*, 542–546.

BOIVIN, S., AND GAGALOWICZ, A. 2001. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *SIGGraph-01*, 107–116.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *SIGGraph-97*, 369–378.

DEBEVEC, P. 1998. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, 189–198.

FAUGERAS, O. 1993. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts.

HAVRAN, V., SMYK, M., KRAWCZYK, G., MYSZKOWSKI, K., AND SEIDEL, H.-P., 2005. Importance sampling for video environment maps.

HDRSHOP. High-dynamic range image processing tools. http://athens.ict.usc.edu/hdrshop/.

HEARN, D., AND BAKER, M. 2004. *Computer Graphics with OpenGL*, third ed. Prentice-Hall PTR, pub-PHPTR:adr.

HEIDRICH, W., AND SEIDEL, H.-P. 1999. Realistic, hardware-accelerated shading and lighting. In *SIGGRAPH*, 171–178.

KAUTZ, J., AND MCCOOL, M. D. 1999. Interactive rendering with arbitrary BRDFs using separable approximations. In *Rendering Techniques '99*, Springer Wien, 247–260.

KAUTZ, J., PAU VZQUEZ, P., HEIDRICH, W., PETER SEIDEL, H., AND UDG, A., 2000. A unified approach to prefiltered environment maps, Sept. 14.

LAFORTUNE, E. P. F., FOO, S.-C., TORRANCE, K. E., AND GREENBERG, D. P. 1997. Non-linear approximation of reflectance functions. In *SIGGRAPH 97 Conference Proceedings*, Addison Wesley, T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, 117–126.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (July), 362–371.

LEWIS, R. R. 1994. Making shaders more physically plausible. *Computer Graphics Forum 13*, 2 (Jan.), 109–120.

MARK, W. R., GLANVILLE, R. S., AKELEY, K., AND KILGARD, M. J. 2003. Cg: a system for programming graphics hardware in a C-like language. In *Proceedings of ACM SIGGRAPH 2003*, J. Hodgins and J. C. Hart, Eds., vol. 22(3) of *ACM Transactions on Graphics*, 896–907.

MERCIER, B., MENEVEAUX, D., AND FOURNIER, A. 2007. A framework for automatically recovering object shape, reflectance and light sources from calibrated images. *International Journal of Computer Vision 73*, 1 (June), 77–93.

MILLER, G. S., AND HOFFMAN, C. R. 1984. Illumination and reflection maps: Simulated objects in simulated and real environments. In *SIGGRAPH '84 Advanced Computer Graphics Animation course notes*. jul.

NELDER, J. A., AND MEAD, R. 1965. A simplex method for function minimization. *Computer Journal 7*, 308–313.

NICODEMUS, F. E., RICHMOND, J. C., HSIA, J. J., GINSBERG, I. W., AND LIMPERIS, T. 1977. Geometric considerations and nomenclature for reflectance. Monograph 161, National Bureau of Standards (US), Oct.

PHONG, B. T. 1975. Illumination for computer generated pictures. *Communications of the ACM 18*, 6, 311–317.

RITSCHEL, T., AND GROSCH, T., 2008. On-line estimation of diffuse materials, Apr. 01.

SHARED-SPACE-ARTOOLKIT. www.hitl.washington.edu/projects/shared space.

SILLION, F., AND PUECH, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco. excellent coverage of radiosity and global illumination algorithms.

WARD, G. J. 1992. Measuring and modeling anisotropic reflection. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, E. E. Catmull, Ed., vol. 26, 265–272.

WU, E., SUN, Q., AND LIU, X. 2004. Recovery of material under complex illumination conditions. In *GRAPHITE*, ACM, Y. T. Lee, S. N. Spencer, A. Chalmers, and S. H. Soon, Eds., 39–45.

XU, S., AND WALLACE, A. M. 2008. Recovering surface reflectance and multiple light locations and intensities from image data. *Pattern Recognition Letters 29*, 11 (Aug.), 1639–1647.

YU, Y., DEBEVEC, P. E., MALIK, J., AND HAWKINS, T. 1999. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *SIGGraph-99*, 215–224.