Modeling Material Reflection with BRDFLab
Adrià Forés\textsuperscript{1,2}, Sumanta N. Pattanaik\textsuperscript{2}, Carles Bosch\textsuperscript{1}, Xavier Pueyo\textsuperscript{1}

\textsuperscript{1} ViRVIG - Universitat de Girona
\textsuperscript{2} University of Central Florida

Technical Report IMA13-01-RR

Dept. Informàtica, Matemàtica Aplicada i Estadística
Universitat de Girona
Modeling Material Reflection with BRDFLab

Adrià Forés
Universitat de Girona
University of Central Florida

Sumanta N. Pattanaik
University of Central Florida

Carles Bosch
ViRVIG - Universitat de Girona

Xavier Pueyo
ViRVIG - Universitat de Girona

Abstract

BRDFLab is a framework for modeling BRDFs intended for people working on representing, measuring and simulating material reflection. We describe the different functionalities provided by BRDFLab, which include the creation, inspection, fitting, and rendering of different types of BRDFs. We also show several examples of applications and describe how to easily expand its functionalities.

1 Introduction

Understanding how light interacts with materials and how to represent such interactions are key aspects in many fields, and particularly in Computer Graphics. Material reflection is typically modeled by means of the BRDF (Bidirectional Reflectance Distribution Function), which describes the fraction of light reflected by the material in a particular direction. BRDFs can be represented in different ways, but analytical expressions are usually preferred for their compactness, easy evaluation, and parameterization.

Over the years, researchers have developed a great variety of BRDF models intended for different needs, ranging from physically-based models, to more empirical or perceptually-based representations. The availability of reflection data has also become more and more common thanks to the development of new measurement devices and image-based techniques. These data may be used to validate analytical models or to study how well they can fit measured data in general. As materials are becoming more and more complex, simulation offers another way to study and developing new reflection models. In this case, the material microstructure is recreated in order to simulate the light scattering within the material model, which helps to understand how the structure affects the reflection of the material. In Figure 1, examples for these three BRDF models are shown.

In all these different situations, there is a need for a general framework able to perform a set of basic operations. This includes visualizing the reflection data, fitting data to analytical models, or rendering objects under different lighting conditions. People working on these topics usually start from scratch to obtain these functionalities, and although some tools have been made available to the community, they only focus on specific parts or are nowadays deprecated.

Aiming at solving this gap, we developed BRDFLab, a general framework for studying, designing and working with reflection models. To develop this system, the following goals have been considered:

- Support different types of BRDFs, including analytical, measured, and simulated materials.
- Interactive visualization of a BRDF in its different forms.
- Fitting from any BRDF to a general analytical expression, supporting different optimizers and metrics.
- Real-time rendering of objects with the BRDF under different lighting conditions.
- Provide tools for performing interactive simulations using graphics hardware.
- Support different ways of extensibility and input/output.

In this paper, we present a system that provides all these functionalities by taking advantage of current programmable graphics hardware. We explain the main techniques used to meet these goals, and we show different
applications of this system. We believe that such a tool
will be valuable for the rendering community for carrying
out further research in the field, for teaching purposes, and
for BRDF modeling in general.

2 Background

The Bidirectional Reflectance Distribution Function
(BRDF) characterizes the directional reflection effects
due to the interaction of light with the material. This
function relates the incoming and reflected light at each point
of a surface as:

\[ f_r(\omega_i, \omega_o) = \frac{dL(\omega_o)}{dE(\omega_i)}, \]

where \( f_r \) is the BRDF, \( L \) is the reflected radiance, \( E \) is the
irradiance or incident light flux per unit area, and \( \omega_i \) and
\( \omega_o \) are the incoming and outgoing directions, respectively.

BRDF functions may be described using different rep-
resentations. Analytical models are mathematical equa-
tions that approximate the reflectance behavior of materi-
als using a set of parameters. Depending on their nature,
these models can be separated into two main categories:
empirical models (such as Phong [16] or Lafortune [11])
or physically-based models (such as Cook-Torrance [6] or
He [10] models).

Materials with complex microgeometry are difficult to
characterize with analytical models. In cases where this
microgeometry is known, simulation can be used to pre-
dict their reflection [23].

Reflectance data can also be measured from real ma-
terials using gonioreflectometers [13, 9]. Although these
data can be later reduced or factorized, the usual approach
is to fit analytical models to them [9, 15].

The development of new material models requires a set
of tools that are rarely available to the community. Ex-
isting tools were mostly designed as a new rendering or
editing technique showcase, and for this reason, they lack
many of the parts required for fully modeling material re-
fection. Rusinkiewicz, for instance, developed a tool for
inspecting BRDF models called \( bv \) [22]. This tool was
very useful at that time, but it was restricted to analytical
models.

Other works focusing on BRDF editing have also been
proposed. BRDF-Shop [5] models BRDFs from the de-
scription of the final material highlights defined onto a
spherical canvas, based on the Ward reflection model.
Ben-Artzi et al. [1] also propose an interactive tool for
editing BRDFs for static scenes and with global illumi-
nation effects. Edition of measured materials is also in-
cluded in this case.

Colbert et al. [4] later presented a technique for real-
time rendering of analytical BRDF models with environ-
ment map lighting. The user is able to modify the model
parameters, visualize a BRDF slice, and render with this
BRDF, but restricted to specific reflection models. Our
system is partially based on this technique for the render-
ing part.

Recently, a system called BRDF Explorer has been pro-
posed for navigating and rendering with analytical and
measured BRDFs [3], which share some functionalities
with our system.

Finally, commercial tools such as Maya, 3D Studio
Max, and LightWave, also provide modeling and render-
ing capabilities for different materials. These systems,
however, are usually restricted to a small set of analytical
models and neither BRDF inspection nor fitting are con-
sidered. Measured materials are only included into some
specific renderers, such as Maxwell or PBRT.

3 System Overview

According to the goals described before, we structured
our system into four main modules: input BRDF defini-
tion, visualization, fitting to analytical models, and data
output. These modules along with the main data work-
flow are shown in Figure 2.

![Figure 2: System overview.](image)

The Input BRDF module encompasses the definition
of an analytical model, the loading of measured data, and
the computation of BRDF data by simulation. Once the
BRDF is available, the visualization step allows the user
to inspect the BRDF by displaying its shape and render
objects with this material. The Fitting module takes an
input BRDF and a set of analytical models and fits the
parameters of these models to the BRDF data, obtaining a
new BRDF representation. This new model can be again
visualized or rendered. The last operation is usually the
output of the obtained BRDF model, data, or even code,
which is done in the Output module.

As one of our goals was to offer extensibility, we de-
sign our system using an object-oriented framework.
Two of the modules especially intended for this purpose
are the Input BRDF and Fitting modules, each of these being represented by a base class that can be easily extended to include new components. Some of these components are, for instance, new BRDF models, simulators, optimizers, and so on. The extensibility part is addressed in Section 5.

In the next section, we describe each of the different modules in detail.

4 System Modules

This section introduces the different parts of our system, explaining how we dealt with our specific goals and the models or tools that are currently supported.

4.1 Input BRDFs

One of the main objectives of BRDFLab was to support different types of BRDF models, including analytical, measured, and simulated models.

**Analytical BRDFs.** Analytical descriptions are very common in the literature and a large variety of models are nowadays available. It is also quite common to combine different models into the same expression, as it provides more flexibility for representing real materials. A classical example would be the use of a Lambertian model for describing the diffuse reflection coupled with a Phong model for the specular lobe.

In our system, we provide two different combination strategies: using channel-independent lobes [15] and using channel-dependent lobes [11]. In a channel-independent configuration, lobes are shared by the different color channels or wavelengths, but each one has channel-specific scaling factors:

$$f_r(\omega_i, \omega_o, \lambda) = \frac{1}{L} \sum_{l=0}^{L} s_{l,\lambda} \cdot f_{r,l}(\omega_i, \omega_o),$$

where $f_{r,l}$ is the reflection model of lobe $l$, $L$ is the number of lobes, and $s_{l,\lambda}$ is the scaling factor or albedo associated to each lobe. In the channel-dependent configuration, the combination uses a different lobe for each channel:

$$f_r(\omega_i, \omega_o, \lambda) = \frac{1}{L} \sum_{l=0}^{L} f_{r,l,\lambda}(\omega_i, \omega_o),$$

where $f_{r,l,\lambda}$ is the reflection model associated to each lobe $l$ and channel/wavelength $\lambda$.

Our system supports a large variety of BRDF models that can be combined using these two expressions, including Lambertian, Blinn-Phong, Ward, Lafortune, Cook-Torrance, and many others. Some of these models are physically-based and others are empirical, while some are isotropic and others anisotropic. Adding new analytical models is easy, as it can be done by simply providing the required expressions, as described in Section 5.

**Measured BRDFs.** Measured data are usually stored as part of a specific database. Incorporating such data into the system simply requires dealing with the specific file format. Two well-known databases are currently supported: the MERL database [13] and the Car Paint database [9]. The input of sparse data as a text file is also supported, which offers a flexible way for adding measured data into the system.

**Simulated BRDFs.** To perform material simulations, we offer the possibility of incorporating data from external applications or libraries. As an example, BRDFLab supports the Scatmech simulation library [7]. This library performs a variety of material simulations over complex material structures, including reflection, transmission, diffraction, and polarization effects.

We also wanted to provide tools for computing simulations inside the system, by taking advantage of the graphics hardware. Such simulations allow the user to change the material properties and/or structure and obtain the corresponding BRDF at interactive frame rates. Given a specific material structure, the simulation performs a ray tracing on the GPU to compute the light interactions with the material in real time. For each incident direction, a set of parallel rays are traced onto the material, computing their bounces and collecting the density of outgoing rays in each direction. By repeating this process for each possible incident direction, we can obtain the four-dimensional BRDF.

Three common material structures are currently supported by BRDFLab: height fields, triangular meshes, and subsurface models consisting of spherical particle distributions (see Figure 3). Each of these models is treated differently for efficiency purposes, using appropriate acceleration structures and specific ray intersection shaders.
For height fields, for instance, we apply a modified ray tracing algorithm based on relief mapping [17], while a more general technique is used for the other cases. The GPU simulation currently assumes that materials consist of perfectly specular micro-facets/particles, so that each light interaction do not add more rays into the simulation. However, a more general approach could be envisaged by using a multi-pass approach and a random walk strategy.

### 4.2 Fitting

The purpose of the fitting step is to approximate a given BRDF - usually measured or obtained by simulation - with a set of analytical models. In most situations, analytical models are preferred, as they tend to be computationally efficient and easy to sample.

The fitting process takes as input the BRDF data, the specified analytical models to be fit, and their initial parameters, also called the initial guess. An optimization process is then applied in order to find the analytic BRDF parameters that match the input data. At each optimization iteration, a new set of parameter values are returned by the optimizer, and the error between the input data and the analytical model is evaluated using those parameters. The optimizer tries to decrease that error at each iteration, until the error threshold or a set of conditions are met.

The analytical models to be fit have to be specified by the user, and any of the combinations introduced in Section 4.1 are allowed. The optimization process can then be started and controlled interactively or used as a batch process, where multiple analytical representations can be sequentially fitted in order to find the best model. In Figure 1-middle, an example of a measured BRDF fitted using a Lafortune model is shown.

We next detail the different strategies, optimizers and metrics supported in our system.

**Strategies.** The system includes both linear and non-linear optimizers, and uses two strategies commonly used in the literature. In the first strategy, only the lobe parameters are optimized by the non-linear optimizer, while the scaling factors are computed at each iteration by a linear Least Squares optimization. As this strategy requires the presence of scaling factors, its use is restricted to a channel-independent combination of lobes (see Section 4.1).

In the second strategy, all the parameters (and scaling factors, if any) are optimized using the non-linear optimizer. This approach supports both lobe combinations.

**Optimizers.** BRDFLab supports multiple non-linear optimizers based on available open-source implementations. Levenberg-Marquardt [12] is one of the optimizers most commonly used in the literature, but we also add support for optimizers such as COBYLA [18], BOBYQA [20], NEWUOA [19], PRAXIS [2], Nelder-Mead Simplex [14], and Subplex [21].

**Error Metrics.** The error metric is the expression that evaluates the difference between two reflection measures during the fitting process. The optimizer tries to minimize this expression for two given BRDFs or reflectance sets.

We decided to support different metrics whether allowed by the optimizers. Some of the metrics we support are the classical $L^2$ metric and the $L^2$ weighted by the incident angle, which are supported by the different optimizers. Another metric used in the literature [15] is:

$$
Error = \sqrt{\sum w |f_{r,s}(\omega_i, \omega_o) \cos \theta - f_{r,t}(\omega_i, \omega_o) \cos \theta|^2},
$$

where $f_{r,s}$ and $f_{r,t}$ are the source and target BRDFs, respectively, and $w$ is the solid angle correction term. This metric is supported by all the optimizers except Levenberg-Marquardt.

As selecting the appropriate method for fitting might be a complex task, we refer the reader to Section 7 for a discussion on the methods that we recommend.

### 4.3 Visualization

The Visualization module encompasses both the display of BRDFs and the rendering of scenes with these models. To provide real-time feedback, these operations are done using the graphics hardware, evaluating the BRDFs through GPU shaders. For analytical models, the shaders evaluate the corresponding expressions and are regenerated on the fly every time the lobe configuration is modified. Measured and simulated BRDFs use 2D or 3D textures to store the reflection data, which will then be accessed by the corresponding shaders.

**BRDF Display.** As visualizing a 4D function is not trivial, it is common to only display the outgoing reflection distribution for a given incident direction. For visualizing this function, we use a geodesic hemisphere representing the outgoing reflection. Each vertex of the hemisphere is displaced by evaluating the corresponding BRDF shader, and the incoming direction can be modified interactively to inspect the model. Other existing operations comprise applying a specific transformation to the data (cube root, log, etc.) or selecting a specific color channel (when dealing with color or spectral BRDFs). Some examples of BRDFs visualized with our system are shown in Figures 1 and 3.

**Rendering.** One of our goals was to support real-time rendering of 3D objects using the current material, so that its final appearance could also be evaluated. To support different lighting conditions, we considered three types of light sources: point light, directional light, and environment lighting.
Figure 4: Real-time rendering under different lighting conditions: aluminum material under a point light source (left); plastic material (middle) and car paint (right) using two different HDR environment maps.

Rendering through point and directional light sources is easily accomplished by means of our BRDF shaders, as a single incident direction needs to be considered at each point. However, dealing with environment lighting may require taking many samples, and a fast importance sampling strategy is then necessary. To support real-time rendering with environment lighting, we decided to incorporate the technique described in Colbert et al. [4]. This technique supports HDR environment maps through a filtered importance sampling approach and the use of Monte Carlo integration techniques. Filtering is accomplished by creating a multiresolution representation of the environment map using a mipmap hierarchy, that is later sampled according to the BRDF probability density function (PDF). Samples generated in the low gradient areas of the PDF access higher mipmap levels, while samples from high gradient areas will access the lower levels. We extended this method to support all of our different analytical models and our two combination strategies, which work by splitting the number of samples according to the current BRDF lobe configuration. In Figure 4, we show some examples of materials rendered with these approaches.

4.4 Output

The main functionality of the Output module is to store the data and models created within the system, so that they can be easily retrieved in other sessions or by external applications. The basic output format is based on an XML file format, which contains the description of the BRDF and the fittings performed on it. Other fitted models extracted from the literature can also be incorporated into this file. This information can later be used into the system to evaluate the different fittings, either by means of visual comparison, rendering, or similar.

Other functionalities that are currently available include exporting the reflectance data, such as the one obtained by a simulation, or exporting the shader code of a specific BRDF model. The later is especially useful for incorporating the obtained material model into an existing rendering pipeline.

5 Extensibility

Offering good extensibility options is crucial for the spread of our system within the community. Its object-oriented design was especially intended for this purpose, but we offer other ways of extensibility as well. In this section, we focus on the two main parts of the system where extensibility is of especial relevance: adding new BRDF models and expanding fitting functionalities.

Adding new reflection models can be accomplished by deriving a new object from one of the existing classes. Under a main BRDF class, a specific class is available for each type of BRDF: AnalyticBRDF, MeasuredBRDF, and SimulatedBRDF. Each object under AnalyticBRDF implements a specific lobe model that can be automatically combined with the existing models. The MeasuredBRDF subclass, on the other hand, deals with a specific material database or format. For simulations, the subclasses serve as interfaces for external simulation libraries, while a specific subclass called GPUSimulatedBRDF includes the interactive simulations within the system.

A similar class called Fitting can be used to incorporate new fitting capabilities. These capabilities mainly include new optimizers and metrics.

For adding new analytical models, we decided to provide an easier solution without the need of modifying the system. In this case, an XML file can be provided with the description of the model equation and its parameters in a formal language. We also require small pieces of shader code for rendering purposes, as automatic translation is not currently supported. This includes code for computing the PDF and its importance sampling, for instance. The analytical models currently supported by the system were incorporated in this way.

6 Applications

We now show some examples of applications of our system, demonstrating its potential in different situations related to material modeling and simulation.

6.1 Image-based material acquisition

Measuring reflection data is a time consuming task that often requires complex setups. In order to easily acquire reflection data, we were recently working on an image-based acquisition process using the existing BRDFLab functionalities. Given a set of images of an object from different viewpoints, our method was based on extracting a sparse set of reflectance data for a selected material. Our system was first extended with an extra module to inspect the images and select the source material. After extracting the reflection data, the model was incorporated
into the pipeline using a new MeasuredBRDF class. Then, we were able to display the measured data, fit analytical models and finally render new objects using the obtained model. Two examples of captured materials are shown in Figure 5.

6.2 BRDF simulation for grooved surfaces

People at the Rochester Institute of Technology were interested on identifying a reference standard to characterize and calibrate measurement systems [24]. As a possible standard, they wanted to consider glossy aluminum surfaces with regular machined grooves. Fabricating surfaces with different groove configurations would have been very costly, thus simulation was preferred to predict the corresponding distribution of reflection light. A new GPU simulation class was added into BRDFLab to simulate grooved materials from a set of parameters: groove width, distance, orientation, etc. Each configuration generated a height field that was then used to simulate light interactions and obtain the corresponding BRDF (see Figure 6). The data was inspected and finally saved to be accessed by an external application. The standard was not finally considered as the manufacture techniques were not able to produce samples with enough repeatability, but the simulations proved to be useful in studying their reflection patterns.

6.3 Fitting measured data

The approximation of measured data to analytical models has been the functionality most commonly used by BRDFLab users. As an example, people at the Technical University of Braunschweig, Germany, used our system to fit their own measured data to different analytical models. They extended our system with new analytical models using our XML-based formal description file. The measured data was then fitted to these models using the Levenberg-Marquardt optimizer. For the final rendering, the PBRT physics-based renderer was used for synthesizing high-quality images. As our renderer does not currently support global illumination effects, exporting our models to a renderer like PBRT is an option that could be incorporated in the future, as explained in Section 9.

7 Evaluation of fitting techniques

As one of the main parts of our system is the fitting module, we next provide some insights on using the different supported optimizers, metrics, or models. We evaluated these options using several measured materials with different gloss properties. These materials were obtained from the MERL database: red fabric, orange paint, pearl paint, and blue metallic paint.

For fitting reflection data, Levenberg-Marquardt (LM) is one of the most well-known optimizers [15]. Although a global solution is not guaranteed, it tends to find an acceptable approximation in most cases. Some of the other methods provide better approximations in some cases, especially BOBYQA and Subplex. However, LM generally provides the best compromise between quality and speed. A comparison is shown in Figure 7 top using Lambert and Ashikmin-Shirley lobes. The fitting quality for one of these materials can also be appreciated in Figure 8.

To evaluate which analytical models fit better to measured data, we considered standard reflection models such as Blinn-Phong, Ward, Lafortune, Cook-Torrance, and Ashikmin-Shirley. From these, we found that Blinn-Phong tends to give the worst results, while the latest three result in better approximations. This can be observed in Figure 7 bottom, using LM as the optimizer. These results agree with similar works on fitting measuring data [15].

Most of our tests have been done with two lobes, using Lambert to represent the diffuse component. Adding more lobes generally decreases both the performance and the accuracy. A better solution is to fit one lobe at a time and subsequently add the other lobes, which results in a better convergence. This option is already contemplated in the system.

For most of our tests, we also use a linear combination of lobes with scaling factors, where the factors or albedos are first fitted with the linear solver. Fitting all the val-
Figure 7: Performance of fittings using different optimizers (top) and analytical models (bottom). Time is in seconds.

Figure 8: Pearlescent paint fitted with different optimizers. In clockwise order: original data, Levenberg-Marquardt, COBYLA, BOBYQA, NEWUOA, PRAXIS, Nelder-Mead Simplex, and Subplex.

ues with a non-linear optimizer may become unstable and slow, and we do not recommend this option. Without scaling factors, i.e. using a channel-dependent combination of lobes, the only model that is suited for fitting is Lafortune, as it preserves energy even without the scaling factor.

Evaluating the different metrics is not easy, as the errors cannot be directly compared. Even by visual inspection of the BRDFs or the renderings, it is not clear which metric performs better. Some metrics give better fittings in some cases while in other cases they perform worse. More tests should be done in this case, using perceptual metrics to evaluate the results.

A final issue is the correct setting of the initial values for the parameters. Each parameter usually comes with a default value and lower/upper bounds. In general, we obtain good convergences with these values when the other considerations are followed.

8 Implementation and Performance

BRDFLab is written in C++ and GLSL and uses several Open Source libraries, which makes it easily portable to several platforms. Current libraries include Qt for the interface, OGRE for rendering, or levmar and NLopt for fitting, among others. The system is Open Source and can be freely downloaded from: http://brdflab.sourceforge.net.

Concerning its performance, the most time consuming part is the fitting step. The time required to find a good estimate may depend on the material, the selected optimizer, or the number of analytical lobes and parameters. However, for simple cases using two or three lobes, good approximations may only take a few seconds. This is especially true for Levenberg-Marquardt (see Figure 7). Visualizing and rendering with a BRDF is usually done in real-time. On a PC with Xeon dual core and a Nvidia 8800 GTX graphics card, displaying and rendering with a point light source was done at 400 fps, while rendering with environment lighting was at 50 fps with 100 samples per point. Finally, simulation time depends on the model and the number of incoming and outgoing rays, but recomputing a BRDF for a different incoming direction usually takes around 3-6 fps.

9 Limitations

Our system is continuously growing but there is still much room for improvement. Simulations with the GPU are currently limited to perfectly specular microgeometry. We should consider more complex light scattering, and wave optics be included for simulating diffraction and other wave-related phenomena [8].

Rendering with environment maps is currently restricted to analytical models. Measured and simulated data can still be rendered with point light sources, but rendering these models with environmental lighting is a remaining task. A more physically-based rendering would also be desirable, including global illumination effects.

The comparison of different fittings is currently done by evaluating the obtained values or by visual examination of the BRDFs and renderings, but perceptual metrics should be developed and included. An interesting possibility would be to use similar metrics during the fitting process, in order to focus on the perceptible differences instead of the physical reflection values.

We would also like to provide more control on the accuracy of the different tasks, which would be especially useful for researchers and engineers working with this system.
10 Conclusions and Future Work

BRDFLab aims to become a general framework for modeling material reflection. The tools already provided for creating, visualizing, and fitting BRDFs were a first step towards this purpose. The interest raised by researchers and groups working on this topic encouraged the development of this framework, and we expect it to continue growing with their significant contributions. We believe that this tool will be helpful for everyone working on BRDF measurement, simulation, and material modeling in general. We also foresee its use as a teaching tool to understand how light reflects from materials and how BRDFs are used to model these effects.

Improving the limitations stated above is one of the next steps to be done. Other directions for future work are the introduction of more general descriptions of analytical BRDF models, such as models based on the combination of different microfacet distributions, Fresnel factors or shadowing terms; the introduction of other representations, such as spherical harmonics or wavelets; and its extension towards the modeling of other light scattering effects, such as transmittance (BTDF) or subsurface scattering (BSSRDF).

References