Painting surface characteristics

Pierre Poulin
Dept. I.R.O.
Université de Montréal

Alain Fournier
Dept. of Computer Science
University of British Columbia

Abstract. Surface illumination proceeds according to the strict rules of the reflection model, the light characteristics and the geometry of the scene. To get a given illumination effect on a surface in a fixed geometry, a user must determine the surface characteristics that will produce this effect. This process is part of what is called *inverse illumination.* In most current modeling systems, the user must rely only on intuition to perform this inverse illumination, which can lead to many modeling/rendering cycles to achieve a satisfactory result.
In this paper, we are concerned with the case where lighting effects are not merely a consequence of the geometry, but rather part of the design. We therefore concentrate our efforts on controlling the surface characteristics and present a tool which can reduce considerably the problem of inverse illumination by using a *painting* paradigm. An interactive system is provided where the user simply applies color points on a surface. The system then attempts in near real time to find the *best values* for the surface characteristics such that the points will retain their assigned color in the final rendering. Depending on the number of constraints (color points) given by the user, the solution is presented as a non-linear constrained optimization and a constrained weighted least-square fitting. We apply our solution to a simple illumination model using ambient, diffuse and specular components to illustrate our approach.

1 Introduction

When designing a scene to be rendered, a user generally first defines the geometry of the scene and the placement and characteristics of the light sources. Once satisfied with the model, the user then fixes the parameters determining the surface characteristics. All of these values determine the appearance of the surface in the final rendering. The surface characteristics can include surface color, quantity of ambient, diffuse and specular reflection, surface glossiness, degree of anisotropy, ratio of dielectric versus conductor properties, transparency, index of refraction, sub-surface scattering, and many others, depending on the illumination model used. Typically, as the illumination model captures more of the behavior of real surfaces, more parameters are required to control the illumination. Moreover, these parameters are often dependent on each other, making it impossible to assign each parameter individually.

In most current modeling systems, very little information is provided to help the user predict what the resulting illumination will be. The user must then rely on intuition and experience to compute mentally the *inverse illumination*: for a given surface and a given geometry of the scene, which set of values for the radiance of the light sources and surface characteristics will produce the desired illumination. A typical session then consists of estimating these values and then rendering the result, which, depending on the scene complexity, the rendering technique and the level of accuracy needed, can take anywhere from a fraction of a second to hours. With the picture as feedback, the user can return to the modeler and tune the parameters to create the desired illumination effects. This process is repeated until the image corresponds to what is wanted or until the user finally decides, after many frustrating attempts, that this is "as good as it is going to get!"

This paper describes an attempt to provide the user with a tool to directly manipulate the illumination effects. More precisely, we present a system that allows a user to control in near real time the surface parameters by simply painting color points on a surface. Each color point is governed by an equation, the variables of which are the surface property parameters. A system of equations arises from all the color points on the surface. So while the user adds and moves color points on a surface, the modeler attempts to determine values for the surface characteristics such that the color points will retain their assigned values (within a certain acceptable range) when the picture is rendered. Each color point introduces a constraint (or two constraints when a range of colors is provided) in a system of equations. When there are less color points than variables, the resulting under-constrained system is solved via non-linear constrained optimization. When there are more color points than variables, the over-constrained system is solved via a weighted least-square approximation with penalty functions to constrain the values of surface parameters. Surface characteristics can therefore be automatically determined as the user adds color points and interactively moves them on the surface.

The system deals only with *local illumination*, that is direct illumination from lights, and therefore each surface can be treated independently of the others. This restriction allows us to deal with complex illumination models by solving non-linear systems of equations. The solution can be computed and displayed in real time on current graphics workstations, providing direct feedback to the user.

This approach reduces the number of inverse illumination tasks (or more exactly *guesses*) that a user must perform, and should lead to an important reduction in the number of modeling/rendering iterations necessary to achieve a given result. It also does not require the user to know all the specifics of an illumination model and prevents the use of *magic* numbers. Since this approach can be used in conjunction with traditional systems, any user, whether experienced or not, should benefit from the extra help provided by this type of modeler.

In the next sections we will first review other work dealing with inverse illumination, then describe the scenario for using our system, and explain how we solve the inverse illumination problem by formulating it as an optimization problem or a fitting problem. Finally we discuss possible extensions of our technique and conclude.

2 Previous work

In most current modeling systems, the user is given a set of valuators to select a value for each available illumination parameter. For almost all renderers assigning a value to a parameter does not result in a prompt update of the image. This approach is typical of the *direct illumination* paradigm commonly used today. Inverse illumination consists of finding automatically values for these parameters. It has been considered a few times before in some different contexts. We summarize here the three systems we know of.

Poulin and Fournier [1] present a system where a user can control the definition and position of a light source by manipulating highlights and/or shadow volumes produced it. Their system uses simple geometric constraints which lead to a unique solution in real time. The only surface characteristic used is the surface glossiness (the exponent in *Phong-type* illumination model) defining the size of a highlight, but it provides no information about the gradient of color in the highlight.

Schoeneman *et al.* [2] address the problem of selecting lights and surface reflectances in the context of a global illumination computation with diffuse reflectors. They use inverse illumination to determine the emitted radiances of the lights. In their system a reflectance must first be associated with each surface. The user then paints colors on surfaces in the scene with a tool simulating spray paint. They assume ideal diffuse surfaces and the light radiances are determined only accounting for direct illumination.

Kawai *et al.* [3] also use inverse illumination for global illumination in a diffuse environment. They use unconstrained non-linear optimization to find a local minimum of a possibly complex objective function. The function includes physical terms (emission, directionality and distribution of the light sources and element reflectance) and terms based on human perception (impression of clearness, pleasantness, and privacy based on the scene brightness). In order to reduce the number of free variables, the user must select the active variables and impose constraints on them.

Inverse illumination problems are of course part and parcel of computer vision. The specific problems investigated include *shape from shading* direction of illuminants from images and identifying surface characteristics from a single or a series of images. The degree of difficulty of these problems is mostly due to the lack of knowledge about both the geometry and the illumination of the scene, but many of the methods are relevant to the problems we deal with here. Fortunately for us, in a modeler for computer graphics the viewing parameters and the exact scene geometry are known and therefore many problems become easier to solve. Our situation is also different from the one found in computer vision since we want the user to control interactively the illumination of a surface. As such we expect the user to provide options and feedback to the system and thus resolve ambiguities when they arise.

3 Painting scenario

Our painting system is built within a traditional modeler. The user can create and position objects and lights, and can set the viewing parameters. The user can set the surface characteristics and light radiances directly. The modeler also provides direct access to the scene geometry. When the user picks on screen a pixel corresponding to a point on an object, geometric information such as its surface normal, and illumination information such as the number and location of lights illuminating that point are available.

The interface to the painting system is simple. A user selects colors and applies them to any visible point in the 3D scene. Each time a color point is applied on the surface, the illumination parameters are recomputed given its color points and the resulting illumination is displayed. A color point is represented on the surface by a small disk. The disk is painted with the selected color but is not shaded. These color points can be added, moved and deleted, their color can be changed and each disk can be scaled up or down. Figure 1 shows some color points and their associated disks on a surface. One can see how the color on each disk is consistent with the color gradient on the surface.

The painted color points thus control the illumination of the surface, providing a high level basis (the illumination model itself) for the interpolation of the surface colors. This is to be contrasted with a traditional 2D painting system where the user can assign a different color to each pixel, or a pattern of colors to a region of pixels. Thus the user has complete control over the final look of the image, but has also full responsibility for illumination *coherence*, which can require the skills of a master painter. To facilitate that task, Hanrahan and Haeberli [4] developed a 3D painting system where a user can paint specific values of surface characteristics directly on points or regions of a 3D model. Our painting system goes further, while making things easier for the user, since it deals with a higher level abstraction than the parameters (color or other surface characteristics) at a given pixel. When the painting is done, the system tries to establish the *best* values for the illumination parameters of each surface. By *best* we mean that the system attempts to optimize certain functions (in the under-constrained case) or to fit values (in the over-constrained case) so the color points will remain as close as possible to their assigned colors when the full rendering is completed. This will be explained in more detail in the following sections. It is important to note that the user does not have to be aware of either the illumination model or the numerical methods used to provide the answers, but has only to judge if the result is acceptable.

If for some combination of painted color points no value for the surface characteristics will satisfy all the constraints, the user can then ask the system to provide colors for a set of given color points for which a solution exists. To do so, the system first relaxes a little the constraints around the latest color point given. If this does not lead to any solution, the constraints are relaxed even more. Colors for which a converging solution can be found are indicated in the color tool.

Now that we have described the general behavior of the system, we describe in the next sections the numerical algorithms used to find the best values for the surface parameters as determined by the painted color points. We will see how each color point can be interpreted as an entry in a system of equations, a constraint in an optimization problem or a sample in a fitting problem.

4 Painting: solving a system of equations

Recall we assume that the geometry of the 3D scene is known and that all the light sources are positioned with known emitted radiance. When a surface element is illuminated, the radiance directly reflected towards the viewer is a function of the illuminants, the scene geometry and the surface parameters. Consider the following popular reflection model [5] applied to a surface element illuminated by m lights:

$$\begin{aligned} L_{pixel} = {} & k_a(\lambda) L_a(\lambda) + \\ & k_d(\lambda) \sum_{i=1}^{m} \int_{\omega_i} (\mathbf{N} \cdot \mathbf{L}_i) L_i(\lambda)\, d\omega_i + \\ & k_s(\lambda) \sum_{i=1}^{m} \int_{\omega_i} F_s(\lambda) (\mathbf{N} \cdot \mathbf{H}_i)^n L_i(\lambda)\, d\omega_i \end{aligned} \tag{1}$$

where $k_a(\lambda)$ is the quantity of ambient reflection,
$k_d(\lambda)$ is the quantity of diffuse reflection,
$k_s(\lambda)$ is the quantity of specular reflection,
$L_i(\lambda)$ is the radiance of light i,
$L_a(\lambda)$ is the ambient radiance,
$\mathbf{N}$ is the surface normal (unit vector),
$\mathbf{L}_i$ is the direction towards light i (unit vector),
$\mathbf{H}_i$ is the bisector direction between $\mathbf{N}$ and $\mathbf{L}_i$ (unit vector),
n is the glossiness factor (surface roughness),
$F_s(\lambda)$ is the specular reflection function,
ω_i is the solid angle formed by light i.

$F_s(\lambda)$ is a function introduced in the specular term to adapt both to traditional models such as Phong or Blinn, or to more complex models.

If we consider $k_a(\lambda)$, $L_a(\lambda)$ and $F_s(\lambda)$ as being constant, then we have an illumination model similar to most software and hardware renderers in current use. This is important in order to control the illumination while displaying in real time the results on a given machine.

For a given color point, the light due to diffuse reflection can be summed for all m lights as:

$$L_d(\lambda) = \sum_{i=1}^{m} \int_{\omega_i} (\mathbf{N} \cdot \mathbf{L}_i) L_i(\lambda)\, d\omega_i \qquad \text{for } \mathbf{N} \cdot \mathbf{L}_i > 0.$$

And similarly for the specular reflection:

$$L_s(\lambda) = \sum_{i=1}^{m} \int_{\omega_i} (\mathbf{N} \cdot \mathbf{H}_i)^n L_i(\lambda)\, d\omega_i \qquad \text{for} \begin{cases} \mathbf{N} \cdot \mathbf{L}_i > 0 \text{ and} \\ \mathbf{N} \cdot \mathbf{H}_i > 0. \end{cases}$$

Generally speaking, $L_d(\lambda)$ and $L_s(\lambda)$ can be computed for any type of light source, whether it is a point light, a linear light or an area light. It is also possible to compute it while accounting for shadows.

Each color point contributes to a new equation at each wavelength. If there are as many independent equations as variables, the system of equations can be solved and a unique value identified for each surface attribute. Looking at Equation (1) when $F_s(\lambda)$ is constant, we can subsume it into $k_s(\lambda)$ and therefore have three terms $(k_a(\lambda), k_d(\lambda), k_s(\lambda))$ which capture the surface color and the proportion of ambient, diffuse and specular reflection, respectively. With the illumination model of Equation (1), a system of three variables can be determined with only three color points in the following form

$$\begin{bmatrix} 1 & L_{d1} & L_{s1} \\ 1 & L_{d2} & L_{s2} \\ 1 & L_{d3} & L_{s3} \end{bmatrix} \cdot \begin{bmatrix} k_a \\ k_d \\ k_s \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix},$$

where L_{dj} and L_{sj} represent the computed values for the diffuse and specular reflections for the j^{th} color point. If a solution to this system exists, a unique value for k_a, k_d and k_s can be computed. Unfortunately it is unlikely a user will be able to provide the exact colors that would lead to a solution. The next two sections present two interpretations of this problem and their respective solutions.

5 Painting: an optimization problem

We can look at this problem as an optimization problem when there are less equations than unknowns. In this scheme, each color point (its range) is given as a volume in the 3D color space of acceptable colors, thus introducing two inequality constraints in each dimension. Additional constraints are associated with the semantics of the parameters in the illumination model. For instance, no parameter in Equation (1) should be negative. Moreover, some illumination models can put an upper limit on the values of some of their parameters. The combination of all the constraints on a surface can lead to no feasible solution (over-constrained), a unique solution (rare) or an infinity of solutions (under-constrained). In the latter case, to provide the user with a unique solution, we need to minimize or maximize an objective function.

5.1 Objective function

Several objective functions are possible and each can lead to a different behavior for the system. In our current implementation, the choice of objective functions is based on the number of color points and their location. Generally, we proceed as follows. We first minimize the ambient term. Surfaces lit only with ambient light appear flat. So by minimizing this term, we ensure that the surface will not appear flat unless constrained so by the user. Once a solution is found, we bound this minimum on k_a within a narrow band to ensure that k_a remains close to its minimum and then maximize the diffuse term. We thus provide more control of the diffuse reflection which often has the most impact on the appearance of a surface.

We found that this combination of objectives leads to a behavior that is intuitive and simple to understand. However it would be possible to *personalize* the behavior of the system by providing the user with a library of objective functions. The user can then interactively select the objectives and combine them.

A series of frames are given in Figure 1 to illustrate the behavior of our system. We chose a simple elongated ellipsoid because of the wide range of possible color gradients due to the continuous changes of its surface normals. The top left image of Figure 1 shows the solution computed for a single color point applied at the bottom part of the ellipse. The ambient component is zero and a dark shade is computed from the diffuse illumination. When the second color point is added in the region in shadow in the top right image of Figure 1, the surface is reshaded with both ambient and diffuse illumination. A third color point is added in the center left image. It then specifies a value for the specular coefficient. Because this color point is added near the boundary of the highlight, it creates a very bright highlight. By moving it closer to the center of the highlight, this color gradient is reduced and therefore the intensity of the highlight is lower. This is shown in the center right image of Figure 1.

The illumination model of Equation (1) can be solved by a simple constrained linear optimization algorithm. These algorithms have the advantage of converging to the global minimum if such a minimum exists. However most of the even slightly more sophisticated illumination models often introduce non-linear constraints. Consider the following illumination model:

$$\begin{aligned} L_{pixel}(\lambda) = {} & k_a S(\lambda) + \\ & k_d S(\lambda) L_d(\lambda) + \\ & k_s L_s(\lambda) \left[M S(\lambda) + (1 - M) \right]. \end{aligned} \tag{2}$$

This model includes a surface color S and simulates a linear dielectric-conductor ratio ($M \in [0, 1]$) for the specular reflection. It is more complex than the previous reflection model. Also notice that the coefficients of ambient, diffuse and specular reflections (k_a, k_d, k_s) are the same for all wavelengths. Therefore we cannot treat the solution independently in each wavelength. The constraints in this model are non-linear. We input these constraints and their gradients into CFSQP [6], a general algorithm to solve constrained non-linear optimization problems.

5.2 Initial guess

The specular reflection can generate a solution within a very narrow domain (spike). If the initial guess is infeasible for some constraints, CFSQP will attempt to find a feasible starting value but can fail to identify one. If this happens, no optimization can be performed even if a solution exists. To avoid this situation, we developed careful initial guesses based on our knowledge of the domains of each variable and the region in which each color point resides.

Another strategy involves enlarging the domain of certain variables and studying the gradient of the converging solutions to determine an initial guess. For instance, a large value for the glossiness factor n produces sharp and small highlights. For a color point to contribute to the highlight, it must reside within the highlight such that $((\mathbf{N} \cdot \mathbf{H})^n > t)$ for a small threshold t. Outside this highlight, the function $(\mathbf{N} \cdot \mathbf{H})^n$ has a very small gradient. If we start with a small value for the glossiness which produces a larger domain and a stronger gradient, we can use the converged solution as an initial guess to the same system of equations but with a slightly larger glossiness value. Adaptively changing the value of the glossiness can then be done automatically. This approach is more expensive computationally but offers an alternative when no other initial guess seems to lead to a solution.

6 Painting: a fitting problem

As the user adds more and more color points to control the illumination, there comes a time when the color points define more constraints than there are free variables. In this case, the problem can be interpreted as fitting the best approximation for each variable subjected to non-linear constraints. We use a non-linear least-square fitting algorithm, *lmder*, taken from MINPACK and available on netlib. *lmder* is based on the Levenberg-Marquardt algorithm.

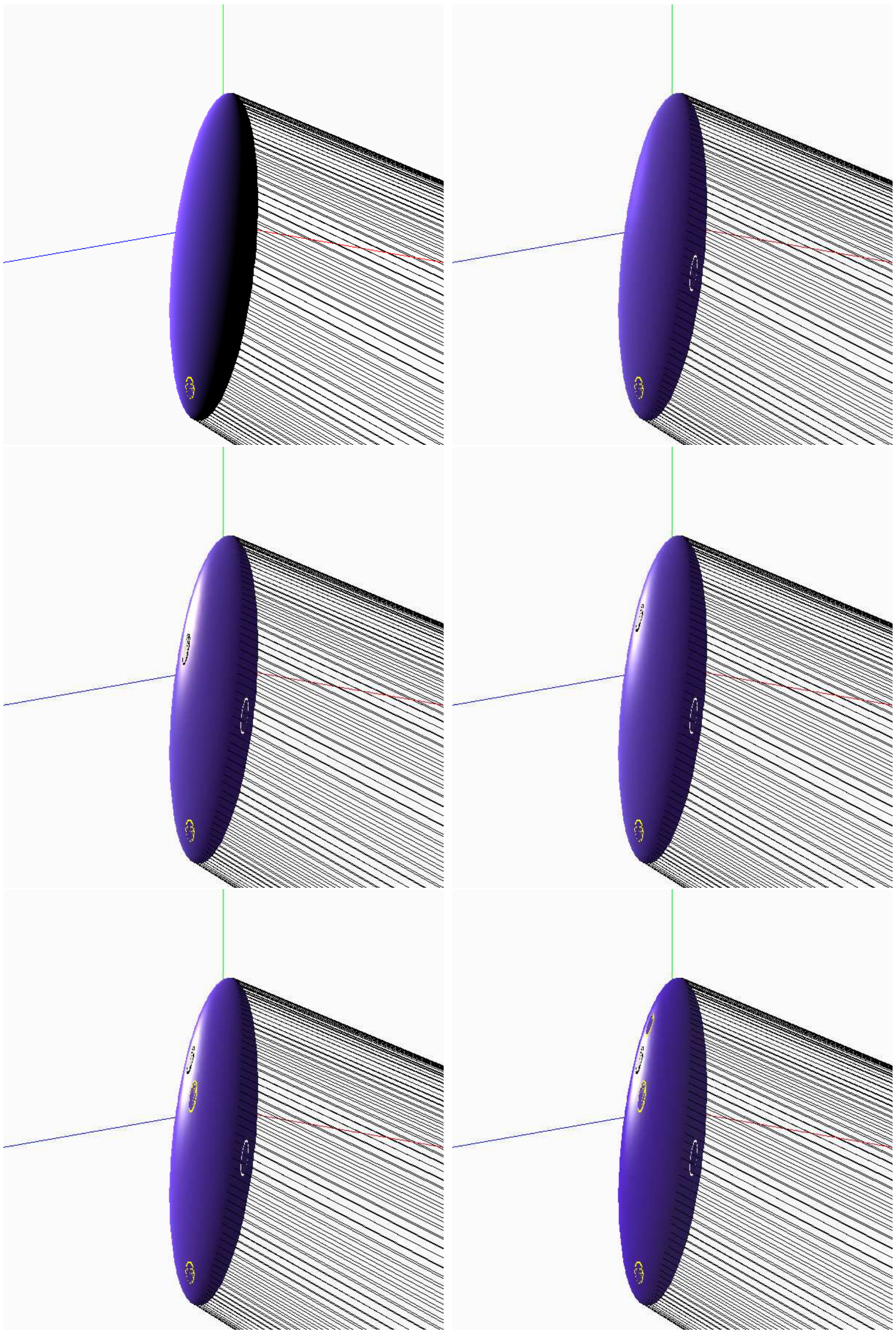

Fig. 1. Painting colors on a surface

In a typical least-square fitting, each variable is free. However in our illumination models, each variable is constrained within a certain domain. To retain this concept of boundaries we use penalty functions to represent these constraints. Let $f_p(x)$ be the penalyzed function of $f(x)$:

$$f_p(x) = \begin{cases} f(x) \cdot e^{bl-x} & \text{if } x < bl \\ f(x) & \text{if } bl \leq x \leq bu \\ f(x) \cdot e^{x-bu} & \text{if } bu < x \end{cases}$$

where bl and bu are the lower and upper boundaries on x, respectively. Any steep function could replace the exponential here. The steeper the function, the higher the penalty will be if a variable goes beyond its domain, but also the stiffer the system will be.

We also modify our system in order to provide different weights depending on the location of the color points. In general least-square problems, a wrong value will have a tendency to drag all the previous values in one direction. Weights can reduce this effect. From illumination Equations (1) and (2), one can observe that a color point within a shadow region on a surface can directly influence very few variables. Therefore we can increase by a large factor (100 in our system) the weight of these points. This ensures that the points in the other regions do not push the ambient term outside its limits. Similarly, the diffuse reflection is usually well controlled by one or two color points. A smaller factor (10 in our system) keeps their contributions important, although less than the ambient ones. Finally, in most cases, the designer uses more color points to finely tune the variation of colors within the highlights. Each such color point will keep a unit contribution. By choosing appropriate weights, it is possible to approximate the behavior of the objective functions in the optimization approach. If done properly, a user will not notice the passage from an optimization problem to a least-square fitting one.

However, unlike the optimization algorithm, the least-square fitting does not guarantee that every color point will retain its color in the final rendering. The weights and penalties do help to keep the final colors as close as possible to the painted ones. This technique offers a nice alternative to the optimization when the user is not able to find an initial feasible guess or when the objective function does not produce exactly the expected illumination.

In the bottom left and right images of Figure 1, two darker color points are added to the previous ones, creating a fitting problem instead of an optimization one. The darker color points positioned near the highlight contour reduce the diffuse illumination while producing a highlight with a sharper gradient.

7 Results and extensions

The main advantage of this combination of optimization and least-square fitting is that their use is mostly invisible to the user. The user does not need to know how many variables there are in the illumination model and what are their contributions to the final illumination. In fact, several illumination models could be used within the modeler and the user would never have to request one over another. The system could adapt itself to the demands of the user when he insists on certain color points to be placed at specific locations, providing a more sophisticated illumination model only when necessary.

We found that for the functions and weights we used, the behavior of our system appeared intuitive, predictable and lead quickly to the desired illumination. If the first painted color points are placed mainly in the shadowed and diffuse regions of a surface, the user gets high control on the final illumination with only a few points. Once these aspects of the illumination are satisfactory, the user can finely tune the look of the highlights with more color points. In many of our test scenes, we found that three to four color points often provide enough control to quickly produce a satisfactory result.

It is important to provide the user with adequate feedback when color points are added or moved. The real-time hardware rendering with the machine illumination model allows one to see directly the surface change as the color points are added or moved on the surface.

From our experience and as Kawai *et al.* [3] indicated, adding more variables to a system usually has the effect of enlarging the domain of possible solutions which makes it easier to converge to a solution. Unfortunately, depending on the contribution of the new variables, they can introduce more local minima. By starting the searches at various locations, the alternative solutions could be presented to the user only when he is not fully satisfied with the current illumination.

We believe that some additional variables could be added to extend the simple illumination models we used here. The first one is the glossiness. We did not consider it because we used the technique in [1] to define the highlights. As previously discussed, a low glossiness value can provide a larger range for initial guesses. It then becomes possible to try to maximize the glossiness in an objective function and therefore to control this variable.

Other parameters from more sophisticated illumination models could be investigated. Some examples include transparency, anisotropy, diffraction, polarization, layered surfaces, etc. However each of these new variables must be studied with respect to the inverse illumination problem which in some cases become quite difficult.

8 Conclusion

Inverse illumination is too difficult a problem to expect the user to solve mentally. In this paper, we presented a system that solves some inverse illumination tasks for the user by using a painting paradigm. Color points are painted on a surface and these points are used to find the best values for the illumination parameters so the painted colors will keep their colors during the final rendering. The color points therefore control the illumination of a surface.

With this technique, it also becomes easier to make small illumination corrections. This technique can also be used to approximate the illumination of real surfaces in the context of mixing real images with computer graphics images by taking sampled colors on a real image of the modeled surface under similar illumination.

When the system is under-constrained we use a non-linear constrained optimization algorithm to find a unique solution for all constraints. The solution guarantees all color points will be within their user-specified boundaries. The objective function defines the behavior of the system. We proposed some combinations of objective functions and techniques to find a feasible initial guess.

When the system is over-constrained we use non-linear least-square fitting to find a solution. The constraints are enforced by penalty functions, weights control the behavior of the system, depending upon the location of the color points. The results are not guaranteed to be within the constraints but are usually very close. A good choice of objective functions and weights and penalty functions provides a smooth transition between the optimization algorithm and the least-square fitting.

Our painting system deals only with direct illumination. By doing so, it provides high control of the full illumination while keeping the number of constraints and variables relatively low. This allows the system to return solutions almost in real time, which used in conjunction with hardware real-time rendering provides a direct feedback to the user adding and moving color points on the surface. Our solution with simple illumination models is a first step that we hope can be tailored to different and more sophisticated illumination models.

Our system can easily be merged with a traditional modeler where the user has to choose directly all the illumination parameters. We offer an alternative so one simply needs to fix or tightly constrain the values of the assigned parameters and let the system find values for the rest.

Our goal in this particular work was to design and implement the tools that will make inverse illumination a practical option for scene modeling. To determine whether and when this approach allows to obtain better results faster is a task that is quite different and remains to be addressed.

References

[1] Pierre Poulin and Alain Fournier. "Lights from highlights and shadows". *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pp. 31–38, March 1992.

[2] Chris Schoeneman, Julie Dorsey, Brian Smits, James Arvo, and Donald P. Greenberg. "Painting With Light". *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pp. 143–146, August 1993.

[3] John K. Kawai, James S. Painter, and Michael F. Cohen. "Radioptimization – Goal-based Rendering". *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pp. 147–154, August 1993.

[4] Pat Hanrahan and Paul E. Haeberli. "Direct WYSIWYG Painting and Texturing on 3D Shapes". *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pp. 215–223, August 1990.

[5] R. Hall. "A Characterization of Illumination Models and Shading Techniques". *Visual Computer*, Vol. 2, No. 5, pp. 268–77, 1986.

[6] Craig Lawrence, Jian L. Zhou, and André L. Tits. "User's guide for CFSQP Version 2.1: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints". Technical Report TR-94-16r1, Electrical Engineering Dept. and Institute for Systems Research; University of Maryland, College Park, 1994.