Interactive Design of Complex Time-Dependent Lighting

Julie Dorsey, James Arvo, and Donald Greenberg
Cornell University

Visualizing complicated lighting sequences while designing large theatrical productions proves difficult. These new techniques achieve fast interaction regardless of scene and lighting complexity, even when used with costly rendering algorithms.

The lighting and stage design of large-scale theatrical productions like operas is tremendously complicated, consisting of hundreds of individual light sources, complex props and sets, projected background scenery, and many elaborate, detailed surfaces. A central aspect of the stage-lighting design process is the time-dependent component.[1] The design must complement such time-dependent phenomena as the motion of performers, the changing focus of attention, and the shifting mood of the plot. In addition, the design must remain precisely synchronized with the music.

The most difficult and time-consuming part of the lighting design process is determining changes in lighting intensity over time, or *dimming*. The lighting designer must determine how lights of different intensities combine at any moment, what temporal changes are necessary to achieve the desired effect, and at what rate the intensity changes should occur. In this context, dimming connotes the complete temporal specification of lights and therefore includes increasing as well as decreasing the intensity.

The complexity and subtleties of the task limit designers to very rough decisions about changes in lighting intensity prior to constructing the final sets and placing and focusing the lighting instruments. Synchronizing a large number of lighting instruments is far too difficult to visualize with conventional media. On the other hand, the simulations shown in the "Results" section depict a scene at several points during a lighting animation. They demonstrate the dramatic effects possible by varying lighting intensities.

Overview of a lighting control system

A lighting control system consists of several primary components. A *light cue* specifies the time interval for the intensity variations of one or more lighting instruments. Organizing lighting changes into cues simplifies the designer's task, but has no other bearing on the actual lighting intensity functions.

Within each light cue are a number of *parts*. A part is a function describing the change in intensity of a subset of lights over time (see Figure 1). In opera and stage-lighting, computerized lighting control boards store this information and execute the cues remotely during performances.

Any collection of lights that vary in unison can be organized into a single *bank*. In this context, lights in a bank are not necessarily juxtaposed on stage; they may occupy any position provided they are all controlled by the same part. For simulation, we can treat each bank as a single distributed light source. In addition, all *fixed* lights—those that remain constant during a performance—can be collected into a single base illumination. The degrees of freedom, n, of a lighting system then equals the number of variable banks.

In current practice, the design of time-dependent lighting effects is an iterative process beginning with *snapshots*, or fixed points, of the lighting scenario. In each snapshot, the designer sets contrasts to focus attention and uses color to convey mood or to establish time of day. After determining snapshots at the start and end of each sequence, the designer is ready to connect them with a series of parts to complete the temporal specification.

A system for interactively designing temporal intensity variations requires several components. First, to qualitatively assess the design concepts, the designer needs to see subtle lighting effects such as soft shadowing, indirect illumination, and color bleeding. These effects are generally only available through costly global illumination algorithms. Second, the ability to quickly preview the design as it changes is of utmost importance, as visualizing all the time-dependent sequences is impossible otherwise. Finally, the animation or playback itself must be sufficiently fast to realistically portray the proposed design.

0272-17-16/95/$4.00 © 1995 IEEE

Previous work in computer-assisted lighting

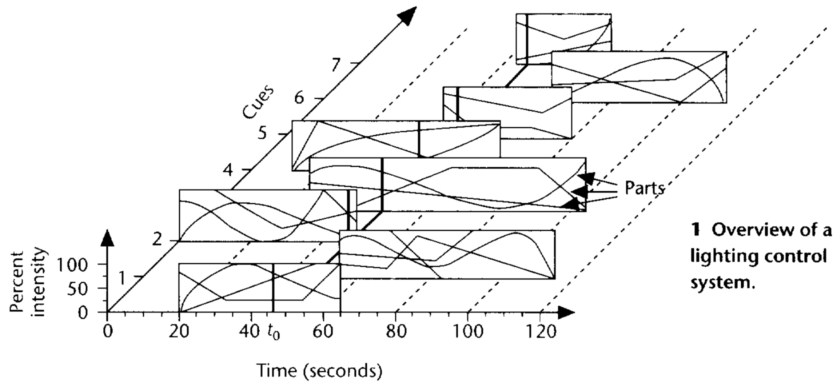
Several previous methods address the problem of adjusting light intensities in the context of global illumination. The radiosity approaches of Chen,[2] George et al.,[3] and Puech et al.[4] update a solution by shooting "negative light" to compensate for changes in lighting or geometry. An approach described by Airey et al.[5] involves storing multiple radiosity values at each vertex and interpolating them as a function of time. Ray-tracing methods, presented by Buckalew and Fussell[6] and Sèquin and Smyrl,[7] demonstrate the possibilities for updating lighting intensities by precomputing and storing view-dependent relationships among lights and surfaces in the environment. Unfortunately, none of these approaches suit theatrical productions because of their complex geometries and numerous lights. Furthermore, the update rates of these approaches, while significantly faster than computing from scratch, are still far too slow for interactivity. By considering a more highly constrained problem—addressing only intensity variations in a fixed geometrical setting—we can take a different approach and overcome these difficulties.

The techniques we introduce here are useful for general lighting design: They provide faster interaction for selecting lighting intensities than previously proposed techniques, and interactivity is not limited by the complexity of the environment or number of lights. In the domain of lighting design, the designers frequently know where to put the lights but not how the lights will combine or how bright to make them. Consequently, selecting appropriate intensities for static lights and scenery and determining intensity variations as a function of time is a useful subproblem of lighting design. We make it our focus here.

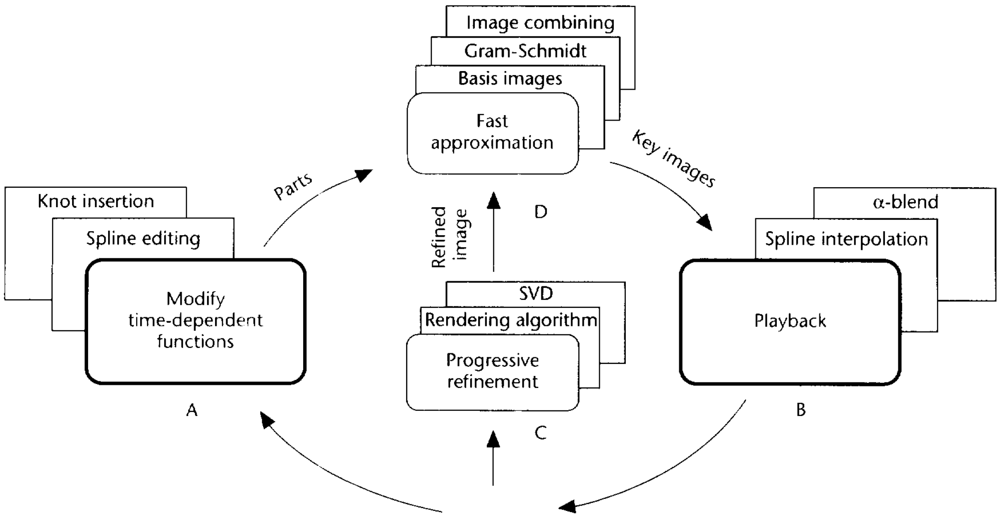
Our approach

In this article we present an approach for interactively designing and previewing variations in lighting intensity. Figure 2 illustrates the main features of our approach, also summarized in the following paragraphs.

- Modification of time dependence. Our approach allows parts composed of cubic splines, enabling the use of sufficiently general light-intensity functions to meet the needs of lighting designers. At any point in time, as the splines are interactively modified, the user receives immediate feedback about the lighting.
- Playback. The current time-dependent lighting scenario can be previewed by a fast playback at any time during the design process. This is accomplished, in part, by exploiting fast α-blending.



1 Overview of a lighting control system.



2 Overview of the interactive design process.

- Progressive refinement. To maintain fast interaction with the user, our approach performs global illumination incrementally and at different levels of accuracy as the design progresses. It uses the rendering algorithm efficiently by (1) exploiting its linearity and (2) constructing an appropriate basis for the required images through singular value decomposition. This approach works for a wide assortment of rendering algorithms, including global illumination.
- Fast approximation. As the user progressively refines the design, the system accumulates a pool of images depicting the scene under different lighting conditions. This pool provides fast and progressively better feedback on changing designs. Previews are constructed from available images using least-squares approximation augmented by coarse images to fill in missing components.

Linearity of light transport

In this and the remaining sections, we develop a methodology for the design of temporal intensity variations. We begin here by discussing the linearity of global illumination and show how to exploit it for our application.

The G operator

Global illumination algorithms model the transfer of radiant energy at the level of geometrical optics, ignoring phenomena such as interference and re-emission of absorbed light. Consequently, the distribution of light existing at radiative equilibrium can be solved independently at each wavelength v:

$$S_v = \mathbf{G}_v L_v \quad \textbf{(1)}$$

where $\mathbf{G}_v$ is a linear operator mapping the source term L_v to the unique equilibrium solution S_v. (Henceforth we drop the v subscript and assume that all computations are carried out for three channels.) While the dependence of **G** on the geometrical and material properties of the environment is highly complex, its action on L is linear, expressing the fact that independent solutions can be superposed. That is,

$$\mathbf{G}\left(\sum_{i=1}^{n} c_i L_i\right) = \sum_{i=1}^{n} c_i \mathbf{G} L_i \quad \textbf{(2)}$$

where $c_1, \ldots, c_n$ are constants and $L_1, \ldots, L_n$ are independent source terms. This linearity is also enjoyed by all common approximations to the exact **G** operator, as embodied in global illumination algorithms.[8]

Note that the summation on the left-hand side of Equation 2 entails addition of n source terms, while the summation on the right requires pointwise addition of the resulting solution functions. These functions may encode either radiance over all surfaces in the scene or simply images from a given vantage point. The latter is particularly appropriate for our application for two reasons. First, working with images is precisely analogous to the *ideal views*—the most advantageous viewing positions in the auditorium—commonly used by designers in evaluating complex lighting. Second, images allow for simpler and faster scaling and addition than do full view-independent radiance functions. Thus, for practical reasons, we will operate on images, although the concepts apply equally well to view-independent radiance functions.[9]

In this article we assume that each environment is accompanied by a **G** operator mapping an n-dimensional vector of light-bank intensities to an image of the globally illuminated environment. Thus, **G** encapsulates a rendering algorithm—ray tracing, radiosity, or some other scheme—and the source function is constrained to a finite number of intensity settings associated with a fixed arrangement of lights. Hence, L is an n-dimensional vector. Initially, all we assume about **G** is that (1) it generates images appropriate to the design task, in terms of both the view and effects modeled; (2) it is linear with respect to sources; and (3) it is very costly to invoke. By virtue of the isomorphism that **G** defines between the space of light intensity settings and the space of images, we can attack some of the problems of time-dependent lighting design. (We later introduce one final assumption about **G** to aid in interactivity.)

The role of linearity

Given a static environment and lighting instruments, consider the task of creating a sequence of images depicting the light intensities changing with time. Figure 1 shows one such sequence with several different light intensity settings.

A naive approach to this problem would be to solve for the global illumination at each time step using the light settings specified for that time. This is prohibitively slow, as **G** is invoked for each frame. Furthermore, if you modify the time-variation, you must recompute all affected frames. Clearly, this is not a viable approach for interactive design.

A more efficient approach is to use fewer applications of **G** by exploiting its linearity. Rather than computing a new image for each of many time steps, you can form the images more quickly by combining other images. Given n images capturing the effect in isolation of each bank on the scene, the combined effect of any collection of lights is given by

$$\mathbf{G}L = \sum_{i=1}^{n} l_i \mathbf{G}\mathbf{e}_i \quad \textbf{(3)}$$

where $L = (l_1, \ldots, l_n)$, the collection of settings for n lights, $\mathbf{e}_i = (0, \ldots, 1, \ldots, 0)$ is the ith standard basis vector, and $\mathbf{G}\mathbf{e}_i$ is the global solution for each light at unit intensity. Using Equation 3 can greatly reduce the number of calls to **G** when n is much smaller than the number of time steps. This improves enormously over the naive method because the computation time for the linear combination of n images on the right of Equation 3 is normally insignificant compared to a single call to **G**. Fast playbacks require further elaboration of this idea, however.

Fast playback

Although the use of linearity in the previous section vastly reduces the computation involved in forming the time sequence (once the images $\mathbf{Ge}_1, \ldots, \mathbf{Ge}_n$ are available), most workstations cannot perform the necessary $n - 1$ image operations per frame in real time, as n might be tens or even hundreds. In this section we describe another optimization that further accelerates the playback process, making real-time playback feasible on displays with fast α-blending. To do this we construct a set of *key images* from which all frames of the playback can be created very efficiently. The idea is reminiscent of keyframes used in animation.

The increase in speed results both from reducing the number of image combinations required per frame and from simplifying the nature of these operations. By expressing the combinations as *lerps*, or linear interpolations, we map the problem directly to α-blending. Although the key images each require up to n image operations to construct, they are—barring completely chaotic time variation—far fewer than the number of frames.

Barycentric coordinates

Observe that the vector of time-varying intensities $L(t)$ is a function from $\mathbb{R}$ to $\mathbb{R}^n$. Its image is a sequence of continuous space curves with breaks at times when any of the intensity settings changes instantaneously.

If these curves are extremely ill-behaved, we can do little to construct the corresponding images more economically than described in the previous section ("Linearity of light transport"). However, consider the following situation: Suppose that between times t_1 and t_2 the curve $L(t)$ lies within a low-dimensional subspace of $\mathbb{R}^n$. Specifically, suppose that $L(t)$ lies within the convex hull of the setting vectors $\{L_1, L_2, \ldots, L_m\} \subset \mathbb{R}^n$ where $m < n$. Then each vector on this segment of $L(t)$ is expressible as a convex combination of the hull vertices; the weights correspond to the *barycentric coordinates*. Equivalently, the image for any $t \in [t_1, t_2]$ can be constructed by lerping the key images corresponding to the setting vectors $\{L_1, L_2, \ldots, L_m\}$, which we call *key settings*. Using the key images, $m - 1$ α-blends suffice instead of $n - 1$ more-general image combinations. If every point of $L(t)$ can be covered by the convex hull of a small number of points, then we can quickly construct every image of the sequence. A practical way to enforce this property is to restrict the parts to low-degree spline curves.

Cubic splines

We now consider cubic splines as a specific example of time-dependent intensity functions. Their flexibility and the simplicity of computing their convex hulls make cubic polynomials ideal for this application.

If each of the n intensity functions varies as a piecewise cubic polynomial with respect to time, the entire curve $L(t)$ can be represented as a collection of n-dimensional Bézier curves. That is, $L(t)$ can be broken into a finite number of segments such that within each segment

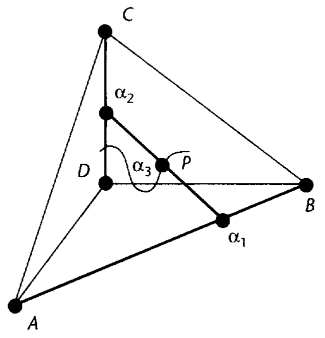
$$L(t) = b_1(t)A + b_2(t)B + b_3(t)C + b_4(t)D \qquad \textbf{(4)}$$

where $b_i(t)$ is the ith Bernstein basis function, with the control points $A, B, C, D \in \mathbb{R}^n$ depending on the segment. The coefficients $b_1(t), b_2(t), b_3(t), b_4(t)$ are barycentric coordinates of $L(t)$ with respect to the control points A, B, C, D. In general, partitioning L into such a collection of segments will require inserting additional knots into some of the parts. Because the Bernstein basis functions are positive and sum to one, the n-dimensional Bézier curve is contained within the convex hull of its four control points regardless of the dimension of the space. Each image I on the curve is then constructed by

$$I(t) = \text{lerp}_{\alpha_3}\left(\text{lerp}_{\alpha_1}(I_A, I_B), \text{lerp}_{\alpha_2}(I_C, I_D)\right) \qquad \textbf{(5)}$$

where $\text{lerp}_t(X, Y) = (1 - t)X + tY$. You can easily see the role of the three interpolations by picturing the points $ABCD$ as forming a tetrahedron in $\mathbb{R}^n$ (see Figure 3).

Clearly, any image corresponding to a point within the tetrahedron can be formed in this way; α_1, α_2, and α_3 depend only on the coordinates of the point, not the nature of the curve it is on. Note that the bound of three blends per frame holds regardless of the number of lights that vary according to independent cubic splines.



3 Interpolation using the tetrahedron.

The collection of all n-dimensional control points of all the Bézier segments comprising L is the set of key settings. Because the corresponding key images are intended for use with traditional α-blending, we must avoid negative pixel values. By inserting additional knots, we can replace all troublesome control points by an equivalent set of positive points, ensuring positivity of the images. Although inserting additional knots increases the number of key images, no more than n of the setting vectors can be linearly independent. So, we need never invoke $\mathbf{G}$ more than n times to generate them all. We pursue this concept further now.

Subspaces of images

To efficiently construct a cubically time-varying sequence using α-blends, we first need a key image corresponding to each control point. In this section, we address the problem of computing the key images as efficiently as possible, again exploiting the linearity of $\mathbf{G}$.

Minimizing global solutions

To illustrate the central idea, we begin by assuming that the n parts are given a priori and address the problem of generating the final images. To generate a given set of key images, it is sufficient to invoke $\mathbf{G}$ only for the linearly independent control points, thus forming a set of *basis images*. We can form the remaining key images from linear combinations of these. If the number of control points is m, then clearly the number of basis images is bounded by $\min(m, n)$. The important question is whether even fewer will suffice.

No a priori reason exists for the parts of any given design to be linearly independent. This implies that there might be a more efficient way of specifying a given design than the one chosen by the user. To allow the user to concentrate on the design, not the efficiency of solution, we describe a technique that automatically finds the minimum number of basis images as well as how to best combine them into the required key images.

Let $\mathbf{L}$ be the $n \times m$ matrix whose columns are the n-dimensional setting vectors given by the m control points:

$$\mathbf{L} \equiv [L_1, \ldots, L_m] \quad \textbf{(6)}$$

To construct all m key images, we require only r basis images where $r = \operatorname{rank}(\mathbf{L})$. Note that $r \le \min(m, n)$. To determine the rank of the matrix $\mathbf{L}$, and to find an appropriate basis for its column space, we perform a *singular value decomposition* (SVD) of $\mathbf{L}$ obtaining

$$\left[\underset{n\times m}{\mathbf{L}}\right] = \left[\underset{n\times r}{\mathbf{Q}}\right]\left[\underset{r\times r}{\mathbf{D}}\right]\left[\underset{r\times m}{\mathbf{R}}\right] \quad \textbf{(7)}$$

where the columns of $\mathbf{Q}$ are orthonormal, $\mathbf{D}$ is nonsingular and diagonal, and the rows of $\mathbf{R}$ are orthonormal.[10] The decomposition $\mathbf{QDR}$ reveals much about the matrix $\mathbf{L}$. First, the rank of $\mathbf{L}$ is determined by the number of diagonal entries in $\mathbf{D}$, which are the *singular values*. The decomposition determines the minimum number of basis images as well as how to combine them into the required key images. Given the SVD, we proceed by invoking $\mathbf{G}$ on each of the r columns of $\overline{\mathbf{L}}$ given by

$$\left[\underset{n\times r}{\overline{\mathbf{L}}}\right] = \left[\underset{n\times r}{\mathbf{Q}}\right]\left[\underset{r\times r}{\mathbf{D}}\right] \quad \textbf{(8)}$$

generating the r linearly independent basis images $\bar{I}_1, \ldots, \bar{I}_r$. The m key images are then formed by using $\mathbf{R}$:

$$\left[I_1 \ldots I_m\right] = \left[\bar{I}_1 \ldots \bar{I}_r\right]\left[\underset{r\times m}{\mathbf{R}}\right] \quad \textbf{(9)}$$

Note that the basis settings $\overline{\mathbf{L}}$ generated by the SVD decomposition will frequently include negative entries, corresponding to negative intensities. This implies that $\mathbf{G}$ must allow propagation of negative energy through the environment and that the basis images might have negative pixels. Consequently, we employ a floating-point format for the basis images,[11] which means that all operations on them must be in terms of "real pixels." The negative values do not appear in the key images, however, as they always correspond to positive settings.

If the lights have been normalized so that at unit intensity each has approximately the same time-averaged influence on the scene, then the magnitudes of the singular values impose a natural order on the basis images. Large singular values indicate a significant overall contribution of the corresponding basis image, while small singular values indicate little contribution. It is common practice to set to zero those singular values that are extremely small relative to the largest values. This allows us to omit images with negligible effect, saving further computation without degrading the visualization. In the next section ("Trading quality for speed"), we make additional use of this ordering.

A simple example

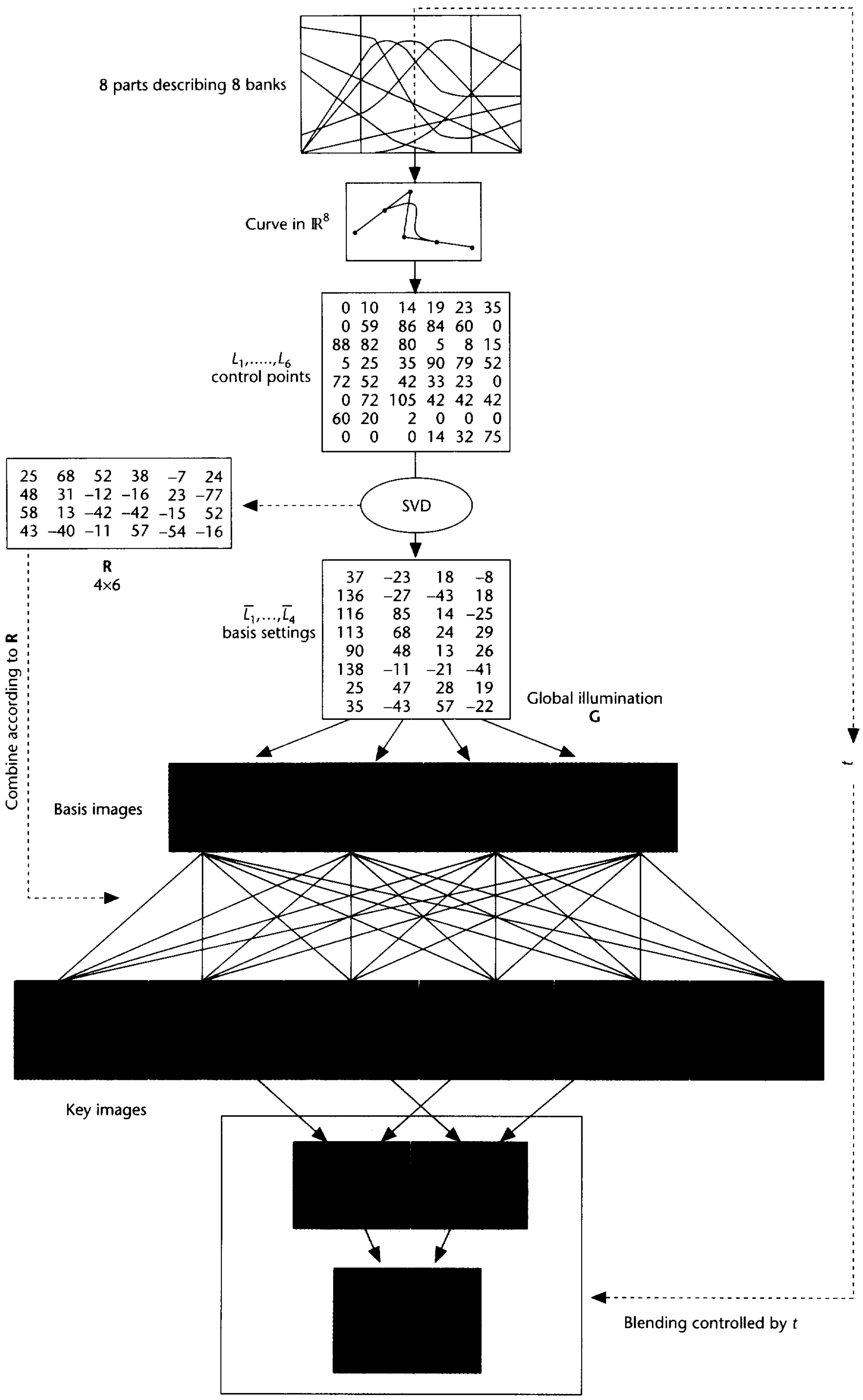
Figure 4 illustrates the steps of the process just outlined. The input consists of eight individual parts, yielding a curve in $\mathbb{R}^8$, describing intensity variations for eight banks of lights. This curve requires six control points, $L_1 \ldots L_6$, to cover its three segments (two of which are linear), corresponding to six key images. The SVD phase reveals that there are only four non-zero singular values and yields the basis settings, $\bar{L}_1 \ldots \bar{L}_4$. Thus, four calls to $\mathbf{G}$ are made to generate the basis images, $\bar{I}_1 \ldots \bar{I}_4$. Note that three of the four basis images contain negative pixels; the images shown in the figure are the absolute values. The basis images are then combined according to $\mathbf{R}$ to obtain the key images, $I_1 \ldots I_6$. Finally, images corresponding to any time t can be computed by blending subsets of four images. Note that the only part of the loop that is time-dependent is the α-blending phase. All other computation is done in a single preprocessing pass.

There is no way to know in advance how much reduction in computation we can expect using the SVD, as each design is different. In practice, we have found that we can significantly reduce the number of solutions by ignoring very small singular values. For example, looking ahead to the scene shown in Figure 6, we used 10 basis solutions to produce 18 key images. For the environment in Figure 8, we used 8 basis solutions to produce 17 key images.

Trading quality for speed

Modifying the algorithm described in the previous section makes it appropriate for iterative design. For example, rather than optimizing the production of a final time sequence, we can introduce incremental updates that can more quickly follow the path the designer takes in exploring possible designs.

The obvious drawback to using SVD is that it requires a complete specification of the parts before you can determine the optimal set of basis images and, therefore, before you create the first image. Yet if the design process itself requires extensive aesthetic judgments in creating the parts, many images must exist before any information is available about the final design. This apparent conflict can be resolved by better meeting the needs of the design process and by adding a bit more capability to the $\mathbf{G}$ operator. At all times the designer must be able to quickly explore a wide variety of design alternatives.


8 parts describing 8 banks
Curve in $\mathbb{R}^8$
$L_1,\ldots,L_6$ control points
SVD
R 4×6
$\bar{L}_1,\ldots,\bar{L}_4$ basis settings
Global illumination G
Combine according to R
Basis images
Key images
Blending controlled by t
t

0	10	14	19	23	35
0	59	86	84	60	0
88	82	80	5	8	15
5	25	35	90	79	52
72	52	42	33	23	0
0	72	105	42	42	42
60	20	2	0	0	0
0	0	0	14	32	75

25	68	52	38	−7	24
48	31	−12	−16	23	−77
58	13	−42	−42	−15	52
43	−40	−11	57	−54	−16

37	−23	18	−8
136	−27	−43	18
116	85	14	−25
113	68	24	29
90	48	13	26
138	−11	−21	−41
25	47	28	19
35	−43	57	−22

4 Overview of the formulation illustrated with a real example.

For this reason it would be ideal to have a separate image per bank before the design process even begins. But computing these n images requires a large time investment—larger than might be warranted by the rank of the final design. The solution lies in the fact that a designer will generally begin with a rough approximation for the placement, content, and duration of the lighting cues, then gradually improve the design until all desired effects have been achieved. During the initial phases, a very rough representation of the preview will usually suffice. High-quality previews become necessary only when the design nears completion. Our improved formulation will reflect this natural progression. The idea is to provide rapid response by sacrificing some of the image quality in the early stages.

The quality parameter

Thus far we have assumed that **G** produces images only of the highest quality, making every invocation very costly. We now relax this assumption by introducing the notion of tunable quality through an accuracy parameter to **G**. In practice, this additional control over **G** depends on the type of rendering algorithm it encapsulates. For example, you can tune a progressive radiosity approach by controlling the meshing or the number of shots. Similarly, a ray-tracing approach can be tuned by controlling image resolution, ray-tree depth, or degree of antialiasing. We will assume that some such control is incorporated into **G**. At a minimum we require two levels of output: approximate but fast, and accurate but slow. We refer to the former images as *coarse* and the latter as *refined*. This added control allows us to construct a design tool that is amenable to progressive refinement and remains interactive at all stages of the design process.

Fast feedback for interactive design

Initially we compute a coarse image for each of the n banks of variable lights and one for the single fixed bank. Once we have done this, the feedback in the initial phase of the design process is very fast. When we need new key images, we can construct them quickly by combining the coarse images corresponding to each of the active banks of lights. With these coarse key images, the real-time playback operates as described in "The role of linearity" above.

Figure 5 shows the interface to an interactive lighting design program. The program allows interactive editing of cues and parts, grouping of lights into banks, and previewing of time-dependent lighting sequences.

Improving preview quality

As the design approaches its objective, we might need more detailed feedback than that afforded by the coarse images. Higher accuracy can help capture subtleties in the lighting, allowing for more detailed evaluation and editing of the design.

To improve the quality of the preview, we need to improve the quality of at least one basis image, such as an image corresponding to a frequently used or important light bank. However, a refined image need not be simply an improvement on an existing image. We can get more benefit from a single new image by simultaneously accounting for many lights. To select an appropriate combination of lights, we use the ordering imposed on the basis settings by the singular values. The vector of settings corresponding to the largest singular value provides the greatest improvement. Additional images can be computed in order of decreasing singular values, although the improvements will be correspondingly smaller.

By viewing the sequence after each new image is added, the designer can evaluate the design at a number of intermediate points. However, we would expect changes to the design as each improvement increases the playback quality. This presents a new problem, as the basis images deemed necessary by SVD will invariably change as the design changes. Fortunately, refined basis images added on behalf of one design can be used for a different but similar design. As we shall see, each refined image will tend to improve and accelerate feedback on subsequent modifications.

New designs from old images

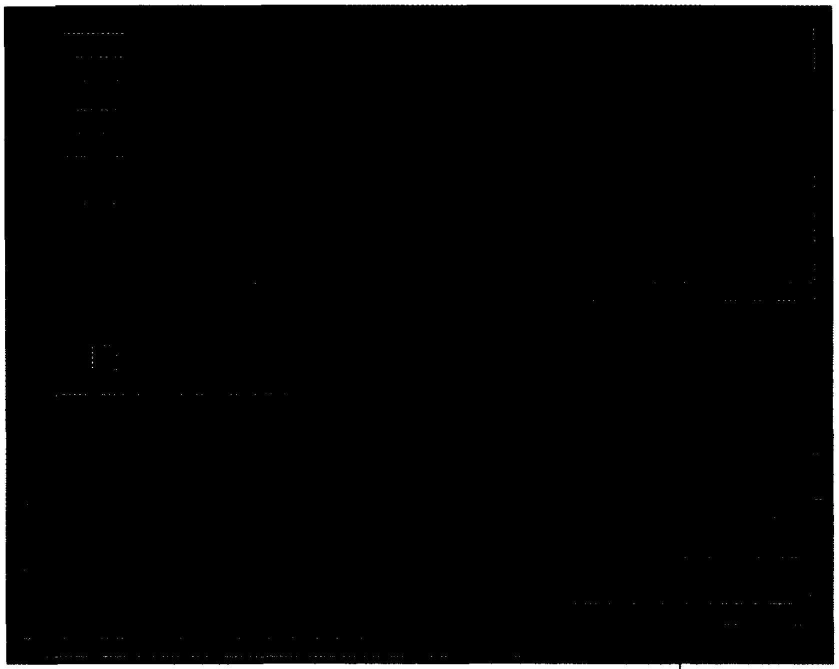
If the design changes after computation of some refined images, the refined images will still be useful, even though they do not match the new basis images suggested by SVD. To see this, imagine the k refined images as the basis of a k-dimensional subspace of $\mathbb{R}^n$. This subspace can be extended to all of $\mathbb{R}^n$ by adding $n - k$ of the coarse images that are linearly independent. New key images created in this way will generally incorporate the refined images to some degree, especially if the new design resembles the old.

At any point in the design cycle, the pool of available images consists of n coarse images and $k \le n$ refined basis images. Initially k is zero, but refined basis images are gradually computed and added to the pool one at a time, as described below. The k refined images always form an orthogonal basis for a k-dimensional subspace, since they are chosen so that each is orthogonal to the subspace spanned by all previous images.

To make maximal use of the refined images, we seek a decomposition of each key image in such a way that priority goes to the refined images. Conceptually, this entails projecting each required image onto the space spanned by the old, then constructing the orthogonal component left unaccounted for from the coarse images. This projection is equivalent to performing a least-squares approximation of the key settings using the refined images in the pool. The Gram-Schmidt procedure[10] is a simple means of computing this approximation and also for filling in the orthogonal component using the coarse solutions. By subtracting the projection of the key setting vector on each element of the pool in descending order of quality (refined first, followed by coarse), the Gram-Schmidt procedure automatically makes maximal use of the refined images in approximating any given image.

SVD and the orthogonal component

In the section "Improving preview quality" we used SVD to select a sequence of images that would most rapidly improve the overall quality of the preview, but we did not take into account previously computed refined images. Doing so requires only a slight modification to the original strategy. Instead of applying SVD to the matrix of settings for the current design, we first remove any components that lie within our existing pool of refined images by subtracting the projection onto the space spanned by the pool. Having removed these components, SVD will identify the most beneficial image to compute in the orthogonal complement of the current refined images. In this way the refined images of the pool always remain orthogonal. This ensures that each new member provides maximum benefit to the current design, never recomputing components obtainable through linear combinations of existing refined images.


5 Screen dump of the interactive lighting design program.

Summary of the design methodology

The design methodology presented in this section fosters interactive design because it gives a designer the ability to get fast feedback while refining a design. It does so by using coarse approximations early on, by progressively refining the previews, and by taking full advantage of all refined images already computed.

We summarize the steps in the design process below:

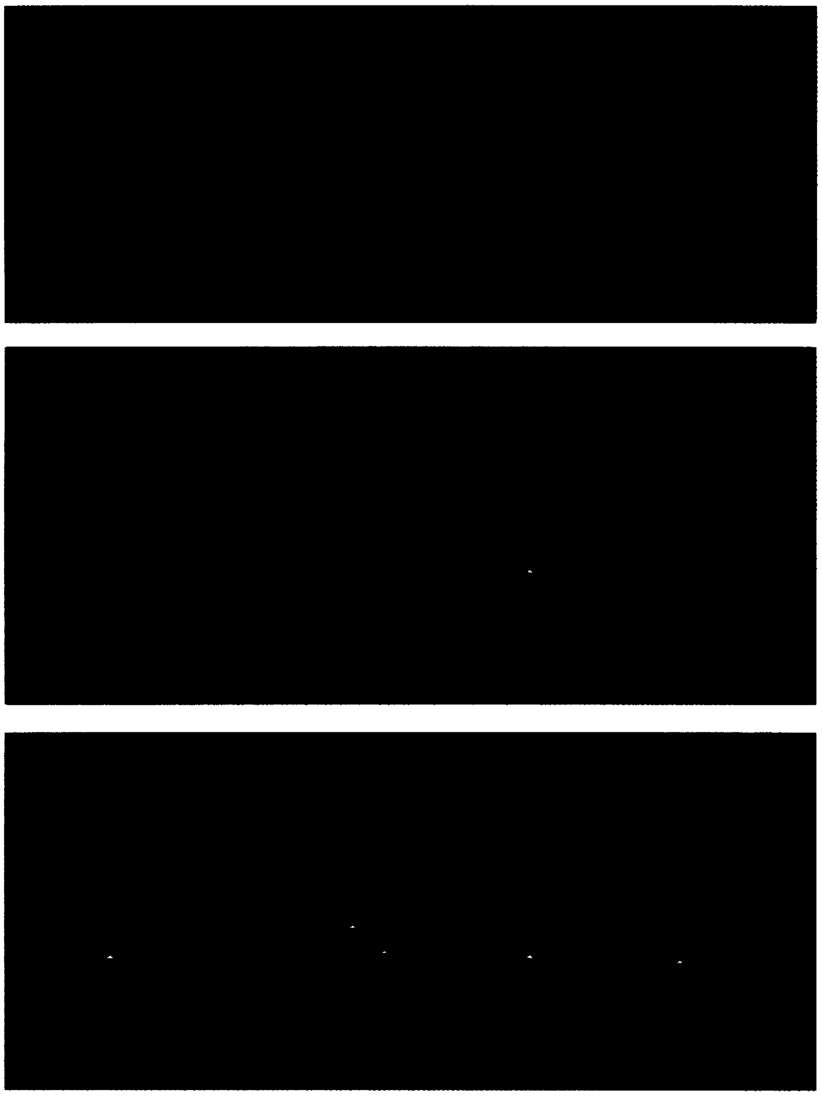
1. Compute a coarse image for each independent bank and one for the fixed bank. This forms the initial pool of basis images for fast playback and also allows for normalization of lights.
2. Interactively specify/modify the parts. Show the effect of changing each bank in real time using the best approximation achievable with the current pool. Recompute the time-averaged intensities of banks that have been modified.
3. Compute the key images for the entire sequence using the current parts and their best approximations achievable with the current pool.
4. Generate a fast playback by α-blending at most four of the key images for each frame in the sequence. If not satisfied with the design, go to step 2.
5. Project the settings of the new design onto the settings of the refined images in the pool using Gram-Schmidt. Apply SVD to the orthogonal component (that which cannot be approximated by the current pool) to determine which new basis images will most rapidly improve the quality of the playback.
6. Use **G** to compute a new refined image corresponding to the largest singular value computed in the previous step. Add it to the pool of basis images and go to step 3.

Results

To demonstrate our techniques, we designed two complex time-dependent simulations based on actual productions from the Metropolitan Opera and rendered with radiosity. Each example involves a large number of time-varying lights. Note an important advantage of our approach: It can provide fast interaction in simulating environments of tremendous complexity, in terms of both the geometry and the lighting effects.

Figure 6 (next page) shows a sequence for Luther's Tavern from *Les Contes d'Hoffmann*. In this scene the time-dependent lighting directs the viewer's attention with sharply focused hanging lamps. Simultaneously, there are numerous changes in the general interior lighting. Gradual transitions from night to day are accomplished by an overall shift from blue to yellow—easily handled with multiple lights of different colors. Another example of changing color is the illumination from the windows.

Figure 7 (next page) depicts a measure of the quality versus computation time required to develop the tavern design. It took a total of 45 minutes to generate the coarse images of the scene—one image for each bank of lights. These images were used for a preliminary design session, shown at the beginning of the chart. Note that the quality level remains low throughout this stage. As the design session progresses, one refined basis image at a time is computed and added to the pool. The quality increases most rapidly with the addition of the first few basis images. Later, the design is modified based on the higher quality preview. A slight drop in quality occurs at these points because the high-quality images now differ slightly from the optimal basis solutions. However, very high quality is maintained because the pool of images is reused to a great extent to accommodate the changes. As the design cycle progresses, changes in the design tend to be smaller, and the larger number of available basis solutions in the pool makes it easier to accommodate these changes. Thus, the slight drops in quality tend to diminish. For this scene, the



6 Time-dependent simulation for the tavern from *Les Contes d'Hoffmann.*

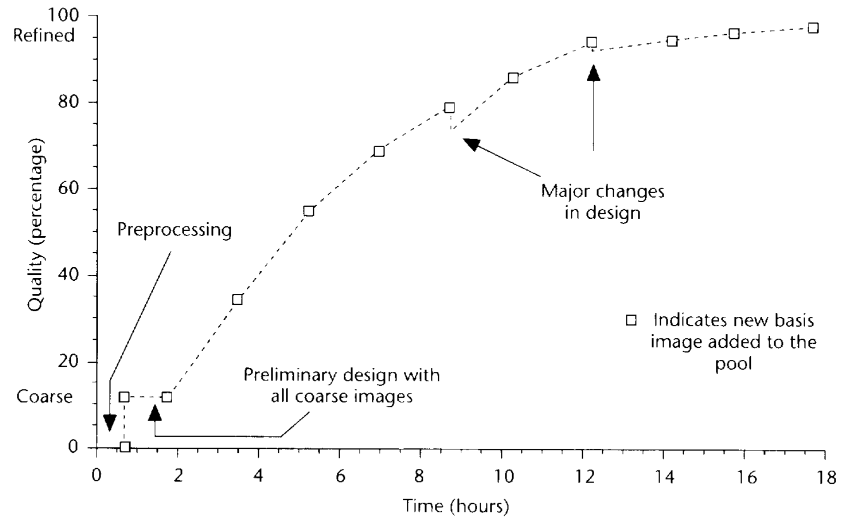
refined solutions took up to two hours each, while the coarse solutions took about three minutes each on an HP 9000/750 workstation.

For the purpose of quality measurements, we computed one refined image per bank before the design session. We then determined the overall quality by comparing the sequence of images at any time against a sequence composed of all refined images. While this data is for just one design session, it is representative of what you should expect from the techniques.

Figure 8 shows a time-varying sequence from *Turandot*. In contrast to the previous example, the time-varying lights in this scene are primarily colored and wash large portions of the set as it transforms from dusk through night. Of particular interest are the interior building lights on the left and right, the red and yellow lanterns in the pavilions toward the rear of the set, and the green lights that bring out the scale figure. The lighting is completed by a transformation of the backdrop from orange to blue. This effect is achieved with two scenic projections[1] and therefore falls within the domain of intensity variations. The refined solutions took up to two and one-half hours each, and the coarse solutions took no longer than four minutes each.

Future work

We described a set of tools for the interactive design and preview of complex, time-dependent lighting intensity variations. By constraining the problem to deal only with intensity variations,



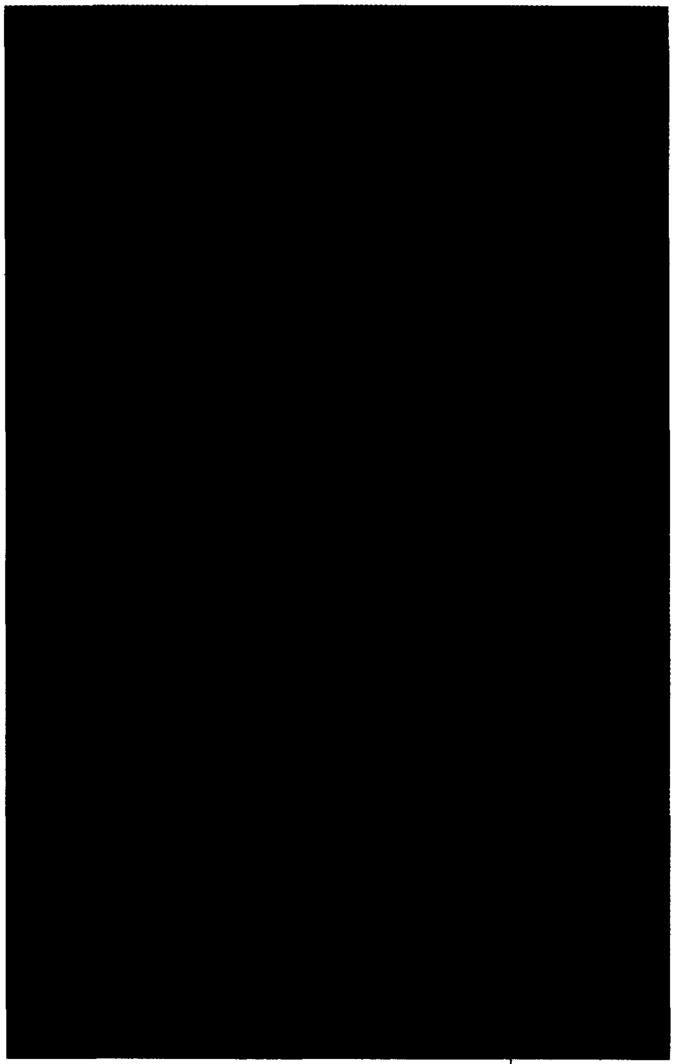
7 Quality versus time for a design session using the tavern scene. Timings were made on an HP 9000/750 workstation.

the formulation can fully exploit the linearity of light transport. Rapid playbacks are accomplished using at most three α-blends per time step regardless of scene and lighting complexity. The technique can use virtually any rendering algorithm and is particularly well suited to costly global illumination algorithms. The number of global illumination solutions required to construct the entire time sequence is minimized using standard linear algebra techniques. The approach supports the iterative nature of design by incrementally following the path the designer takes in exploring possible designs.

We would like to pursue several areas of inquiry. In the current implementation we use two extreme quality levels when computing global illumination. We would like to generalize this to a continuum, although doing so complicates the matter of determining which image is most advantageous to improve at any time. The potential benefit would be faster and more uniform improvement. In addition, we would like to combine the work with scans of real scenes or artistic sketches. This could be a convenient way of bridging the gap between preliminary sketches and coarse computer-generated renderings. Finally, we would like to consider nonlinear problems such as varying the geometry and material attributes.[12] ■

Acknowledgments

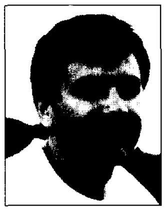
We are indebted to Gil Wechsler and Wayne Chouinard at the Metropolitan Opera for providing documentation of real set and lighting designs and for their input into this research. Hewlett-Packard generously provided the workstations used for the simulations. The Program of Computer Graphics at Cornell University is one site of the NSF/ARPA Science & Technology Center for Computer Graphics and Scientific Visualization. Mark Shepard modeled the *Turandot* environment, and Chaim Zeitz modeled the *Les Contes d'Hoffmann* scene. Thanks to Kurk Dorsey, Hurf Sheldon, Chris Schoeneman, and Brian Smits for their general assistance.


8 Time-dependent simulation for Turandot.

References

1. J.O. Dorsey, F.X. Sillion, and D.P. Greenberg, "Design and Simulation of Opera Lighting and Projection Effects," *Computer Graphics* (Proc. Siggraph), Vol. 25, No. 4, Aug. 1991, pp. 41-50.
2. S.E. Chen, "Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System," *Computer Graphics* (Proc. Siggraph), Vol. 24, No. 4, Aug. 1990, pp. 135-144.
3. D.W. George, F.X. Sillion, and D.P. Greenberg, "Radiosity Redistribution for Dynamic Environments," *IEEE CG&A*, Vol. 10, No. 4, July 1990, pp. 26-34.
4. C. Puech, F. Sillion, and C. Vedel, "Improving Interaction with Radiosity-based Lighting Simulation Programs," *Computer Graphics*, Vol. 24, No. 2, Mar. 1990, pp. 51-57.
5. J.M. Airey, J.H. Rohlf, and F.P. Brooks, "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments," *Computer Graphics*, Vol. 24, No. 2, Mar. 1990, pp. 41-50.
6. C. Buckalew and D. Fussell, "Illumination Networks: Fast Realistic Rendering with General Reflectance Functions," *Computer Graphics* (Proc. Siggraph), Vol. 23, No. 3, July 1989, pp. 89-98.
7. C.H. Séquin and E.K. Smyrl, "Parameterized Ray Tracing," *Computer Graphics* (Proc. Siggraph), Vol. 23, No. 3, July 1989, pp. 307-314.
8. J.T. Kajiya, "The Rendering Equation," *Computer Graphics* (Proc. Siggraph), Vol. 20, No. 4, Aug. 1986, pp. 143-150.
9. C. Schoeneman et al., "Painting with Light," *Computer Graphics* (Proc. Siggraph), Vol. 27, No. 2, Aug. 1993, pp. 143-146.
10. G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins, Baltimore, 1983.
11. G. Ward, "Real Pixels," in *Graphics Gems II*, J.R. Arvo, ed., Academic Press, New York, 1991.
12. J. Nimeroff, E. Simoncelli, and J. Dorsey, "Efficient Re-rendering of Naturally Illuminated Environments," *Proc. 5th Eurographics Workshop on Rendering*, Eurographics, Darmstadt, June 1994, pp. 359-374.

Julie Dorsey *is an assistant professor in the Department of Architecture at the Massachusetts Institute of Technology. Her research interests include realistic image synthesis, interactive simulation, computer-aided design, and lighting and acoustical design. Dorsey received BS and BArch degrees in architecture (1987) and MS (1990) and PhD (1993) degrees in computer graphics from Cornell University.*



James Arvo *is a member of the research staff at the Program of Computer Graphics at Cornell University. His research interests include global illumination and volume rendering. Arvo received his MS and MPhil degrees in computer science from Yale University, his MS in mathematics from Michigan State University, and his BS in mathematics from Michigan Technological University. He is a member of ACM.*

Donald Greenberg *is director of the NSF Science and Technology Center for Computer Graphics and Scientific Visualization. He is also director of the Program of Computer Graphics at Cornell University and teaches the computer graphics and computer-aided design sequence in the department of computer science. He is primarily concerned with research advancing the state of the art of computer graphics and in applying these techniques to various disciplines. His application work now focuses on medical imaging, architectural design, and interactive video.*

In 1987 Greenberg received the ACM Steven Coons Award for his outstanding creative contributions in computer graphics. He also received the National Computer Graphics Association Academic Award in 1989. He is a member of the National Academy of Engineering.

Readers may contact Dorsey at Rm. 5-418, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139, e-mail dorsey@mit.edu.