# Interactive Design of Complex Time-Dependent Lighting

*Julie Dorsey*
*James Arvo*
*Donald Greenberg*

Time-dependent sequence from *Turandot*

Program of Computer Graphics
Cornell University
Ithaca, New York 14853

## Abstract

We describe new tools for the interactive design of complex, time-dependent lighting in scenes with fixed geometry. The work is motivated by the difficulty in visualizing complicated lighting sequences during the design of large-scale theatrical productions. Fast interaction is achieved regardless of scene and lighting complexity, even when used in conjunction with costly rendering techniques such as radiosity and ray tracing. Time-variant lighting is simulated using linear combinations of static images, each depicting the scene under different lighting conditions. Multiple levels of accuracy facilitate interactive design by trading image quality for fast feedback in a controlled way. Coarse approximations are used in the preliminary design stage to allow for instantaneous feedback, then the sequence is progressively refined by computing basis solutions in order of increasing overall contribution. When changes to the design are required, existing global solutions are re-used to the greatest extent possible. The techniques are demonstrated with complex models based on actual stage sets.



**CR Categories and Subject Descriptors:** I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism, Animation; I.3.3 [**Computer Graphics**]: Picture/Image Generation; I.3.6 [**Computer Graphics**]: Methodology and Techniques - Interaction techniques;

**Additional Key Words:** $\alpha$-blending, image compositing, animation, simulation, global illumination, radiosity, ray tracing, lighting design.

# 1 Introduction

The lighting and stage design of large scale theatrical productions, such as operas, is tremendously complicated and consists of hundreds of individual light sources, complex props and sets, projected background scenery, and many elaborate, detailed surfaces. One of the central aspects of the stage-lighting design process is the time-dependent component (see [4] and cited references). 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. This process is commonly referred to as *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 the construction of the final sets and the placement and focusing of the lighting instruments. The synchronization of a large number of lighting instruments is far too difficult to visualize with conventional media. Figure 8, which depicts a scene at several points during the course of a lighting animation, demonstrates the dramatic effects that are possible by varying lighting intensities.

## 1.1 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. The organization of lighting changes into cues simplifies the task of the designer 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

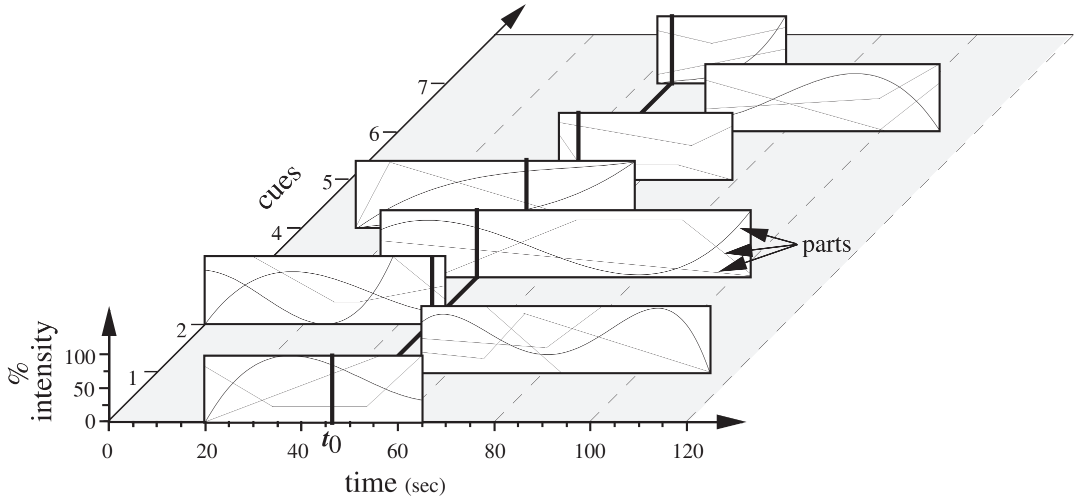


**Figure 1:** Overview of a lighting control system.

change in intensity of a subset of lights over time (see Figure 1). In opera and stage-lighting, computerized lighting control boards store the above information and execute the cues remotely during performances.

Any collection of lights that vary in unison can be organized into a single *bank*.[1] Thus, each light of a bank changes according to the same part. For the purpose of 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 are then 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 in order to focus attention and uses color to convey over-all mood or to establish the time of day. After snapshots are determined 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 the interactive design of temporal intensity variations requires several components. First, to qualitatively assess the design concepts, the designer needs to

[1] In this context, lights in a *bank* are not necessarily juxtaposed spatially, but may be at any position provided they are all controlled by the same part.

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/playback itself must be sufficiently fast to realistically portray the proposed design.

### 1.2 Previous Work in Computer-Assisted Lighting

Several methods have been proposed to address the problem of adjusting light intensities in the context of global illumination. The radiosity approaches of Chen [3], George et al. [5] and Puech et al. [9] update a solution by shooting "negative light" to compensate for changes in lighting or geometry. An approach described by Airey et al. involves storing multiple radiosity values at each vertex and interpolating them as a function of time [1]. Ray-tracing methods, presented by Buckalew and Fussell [2] and Séquin and Smyrl [11], 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 are suitable for the application addressed here because of the complex geometries and large numbers of lights used for theatrical productions. Furthermore, the update rates, while significantly faster than computing from scratch, are still far too slow to allow interaction. By considering a more highly constrained problem—addressing only intensity variations in a fixed geometrical setting—we can take a different approach and overcome this difficulty.

The techniques introduced in this paper are useful for general lighting design in that 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 about where to put the lights but not how the lights will combine or how bright to make them. Consequently, the task of selecting appropriate intensities for static lights and scenery and determining intensity variations as a function of time is a useful subproblem of lighting design, and this is our focus.

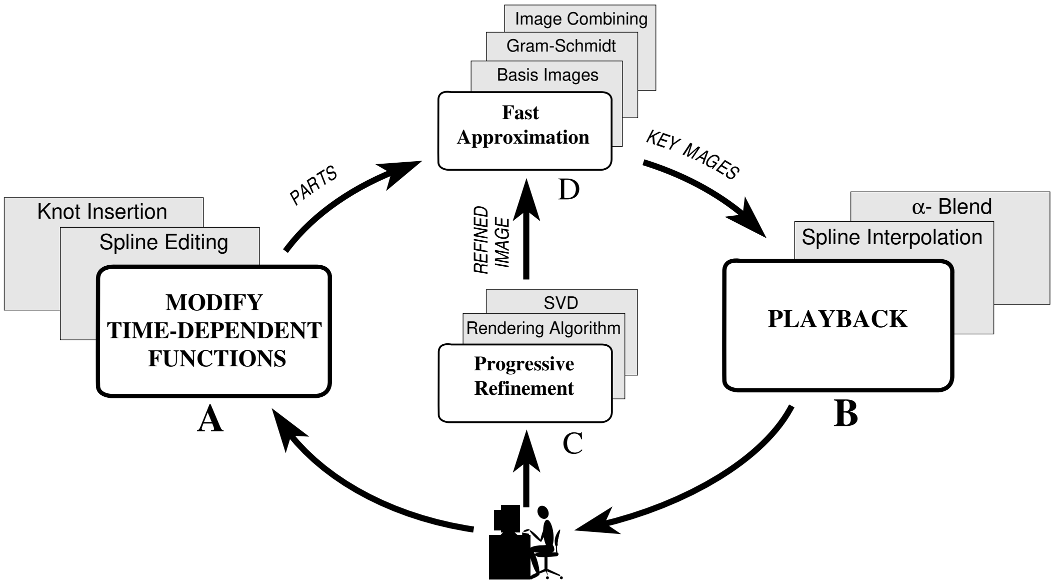


**Figure 2:** Overview of the interactive design process.

### 1.3 Overview of Approach

In this paper, we present an approach for the interactive design and preview of lighting intensity variations. The main features of our approach are illustrated in Figure 2 and summarized below.

**A: Modification of Time-Dependence.** It enables the use of sufficiently general light-intensity functions to meet the needs of lighting designers by allowing parts comprised of cubic splines. Immediate feedback about the lighting at any point in time is given as the splines are interactively modified.

**B: Playback.** At any time during the design process the current time-dependent lighting scenario can be previewed via a fast playback. This is accomplished, in part, by exploiting fast $\alpha$-blending.

**C: Progressive Refinement.** To maintain fast interaction with the user, global illumination is performed incrementally and at different levels of accuracy as the design progresses. Efficient use is made of the rendering algorithm by exploiting its linearity and constructing an appropriate basis for the required images using singular value decomposition. This approach works for a wide assortment of rendering algorithms, including global illumination.

**D: Fast Approximation.** As the design is progressively refined, a pool of images is accumulated 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.

In the remaining sections, we develop a methodology for the design of temporal intensity variations. Section 2 discusses the linearity of global illumination and shows how it can be exploited for this application. Section 3 introduces techniques to facilitate rapid playback of time-dependent sequences. Section 4 presents tools to optimize the computation of global illumination solutions. Section 5 introduces a quality parameter, so that one can trade quality for speed in previewing the lighting animations. Examples of complex lighting designs are also presented.

## 2 Linearity of Light Transport

### 2.1 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 $\nu$:

$$S_\nu = \mathbf{G}_\nu L_\nu \tag{1}$$

where $\mathbf{G}_\nu$ is a linear operator [2] mapping the source term $L_\nu$ to the unique equilibrium solution $S_\nu$. 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

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

[2] Henceforth we drop the $\nu$ subscript and assume that all computations are carried out for three channels.

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 [7].

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 either encode 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* 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 [10].

In this paper we shall 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** is an encapsulation of a rendering algorithm, be it ray tracing, radiosity, or some other scheme, and the source function is constrained to be a finite number of intensity *settings* associated with a fixed arrangement of lights. Hence, $L$ is an $n$-dimensional vector. Initially, all we shall assume about **G** is that 1) it generates images appropriate to the design task, both in terms of 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. In section 5 we introduce one final assumption about **G** to aid in interactivity.

### 2.2 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 the time-variation is modified, all affected frames must be re-computed. 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, the images can be formed more quickly by combining other images. Given $n$ images capturing the effect of each bank on the scene in isolation, the combined effect of any collection of lights is given by

$$\mathbf{G}L = \sum_{i=1}^{n} \ell_i \mathbf{G}\,\mathbf{e}_i \tag{3}$$

where $L = (\ell_1, \ldots, \ell_n)$, the collection of settings for $n$ lights, $\mathbf{e}_i = (0, \ldots, 1, \ldots, 0)$ is the $i$'th standard basis vector, and $\mathbf{G}\,\mathbf{e}_i$ is the global solution for each light at unit intensity. Use of Equation 3 can greatly reduce the number of calls to **G** when $n$ is much smaller than the number of time steps. This is an enormous improvement 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**. Further elaboration of this idea is required for fast playbacks, however.

## 3 Fast Playback

Although the use of linearity in the previous section vastly reduces the computation involved in forming the time sequence (once the images $\mathbf{G}\,\mathbf{e}_1, \ldots, \mathbf{G}\,\mathbf{e}_n$ are available), it will remain beyond the capabilities of most workstations to perform the necessary $n - 1$ image operations per frame in real-time, as $n$ may be tens or even hundreds. In this section we describe another optimization that will further accelerate the playback process, making real-time playback feasible on displays with fast $\alpha$-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 is gained both by reducing the number of image combinations required per frame and by simplifying the nature of these operations. By expressing the combinations as "lerps," or linear interpolations, we map the problem directly to $\alpha$-blending. Although the key images each require up to $n$ image operations to construct, barring completely chaotic time-variation they are far fewer than the number of frames.

### 3.1 Barycentric Coordinates

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

If these curves are extremely ill-behaved, there is little we can do to construct the corresponding images more economically than in Section 2. 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 $\mathcal{R}^n$. Specifically, suppose that $L(t)$ lies within the convex hull of the setting vectors $\{L_1, L_2, \ldots L_m\} \subset \mathcal{R}^n$ 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, $m - 1$ $\alpha$-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 every image of the sequence can be quickly constructed. A very practical means of enforcing this property is to restrict the parts to be low-degree spline curves.

### 3.2 Cubic Splines

We now consider cubic splines as a specific example of time-dependent intensity functions. Their flexibility along with 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

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

where $b_i(t)$ is the $i$th Bernstein basis function, with the control points $A, B, C, D \in \mathcal{R}^n$ 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

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

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

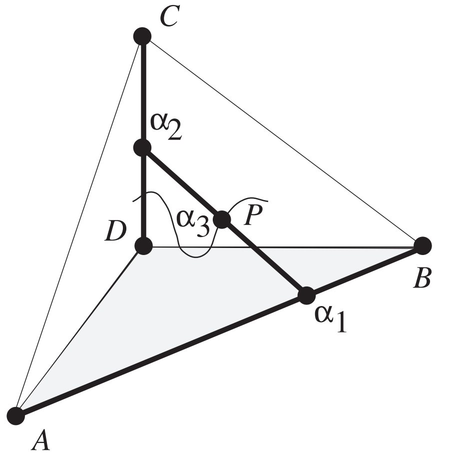


**Figure 3:** Interpolation using the tetrahedron.

Clearly any image corresponding to a point within the tetrahedron can be formed in this way; $\alpha_1$, $\alpha_2$, and $\alpha_3$ depend only on the coordinates of the point and 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.

The collection of all $n$-dimensional control points of all the Bézier segments comprising $L$ is the set of key settings. As the corresponding key images are intended for use with traditional $\alpha$-blending we must avoid negative pixel values. By inserting additional knots all troublesome control points can be replaced 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 **G** more than $n$ times to generate them all. This is pursued further in the next section.

## 4 Subspaces of Images

### 4.1 Minimizing the Global Solutions

To efficiently construct a cubically time-varying sequence using $\alpha$-blends, we first require 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 **G**. 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 **G** only for the linearly independent control points, thus forming a set of *basis images*. The remaining key images can be formed 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.

There is no a priori reason why the parts of any given design will be linearly independent. This implies that there could 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 **L** be the $n \times m$ matrix whose columns are the $n$-dimensional setting vectors given by the $m$ control points:

$$\mathbf{L} \equiv [L_1, \cdots, L_m] \tag{6}$$

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

$$\left[ \begin{array}{c} \\ \underset{n \times m}{\mathbf{L}} \\ \\ \end{array} \right] = \left[ \begin{array}{c} \\ \underset{n \times r}{\mathbf{Q}} \\ \\ \end{array} \right] \left[ \underset{r \times r}{\mathbf{D}} \right] \left[ \quad \underset{r \times m}{\mathbf{R}} \quad \right] \tag{7}$$

where the columns of **Q** are orthonormal, **D** is non-singular and diagonal, and the rows of **R** are orthonormal [6]. The decomposition **QDR** reveals much about the matrix **L**. First, the rank of **L** is determined by the number of diagonal entries in **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 **G** on each of the $r$ columns of $\overline{\mathbf{L}}$ given by

$$\left[ \begin{array}{c} \\ \underset{n \times r}{\overline{\mathbf{L}}} \\ \\ \end{array} \right] = \left[ \begin{array}{c} \\ \underset{n \times r}{\mathbf{Q}} \\ \\ \end{array} \right] \left[ \underset{r \times r}{\mathbf{D}} \right], \tag{8}$$

generating the $r$ linearly independent basis images $\overline{I_1}, \ldots, \overline{I_r}$. The $m$ key images are then formed by using **R**:

$$[I_1 \cdots I_m] = \left[\overline{I_1} \cdots \overline{I_r}\right] \left[ \quad \underset{r \times m}{\mathbf{R}} \quad \right]. \tag{9}$$

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 **G** must allow propagation of negative energy through the environment, and that we must allow for negative pixels in the basis images. Consequently, we employ a floating point format for the basis images [12], 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 those that are small 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 Section 5, we make additional use of this ordering.

### 4.2 A Simple Example

Figure 4 illustrates the steps of the process outlined in this section. The input consists of eight individual parts, yielding a curve in $\mathcal{R}^8$, 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, $\overline{L_1} \ldots \overline{L_4}$. Thus, four calls to **G** are made to generate the basis images, $\overline{I_1} \ldots \overline{I_4}$. 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 $\boldsymbol{R}$ 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 $\alpha$-blending phase. All other computation is done in a single pre-processing pass.

There is no way to know in advance how much reduction in computation might be expected by using the SVD, as each design is different. In practice, we have found that the number of solutions can be significantly reduced by ignoring very small singular values. For example, in 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.

## 5 Trading Quality for Speed

We now describe modifications to the algorithm of the previous section to make it appropriate for iterative design. Rather than optimizing the production of a final time

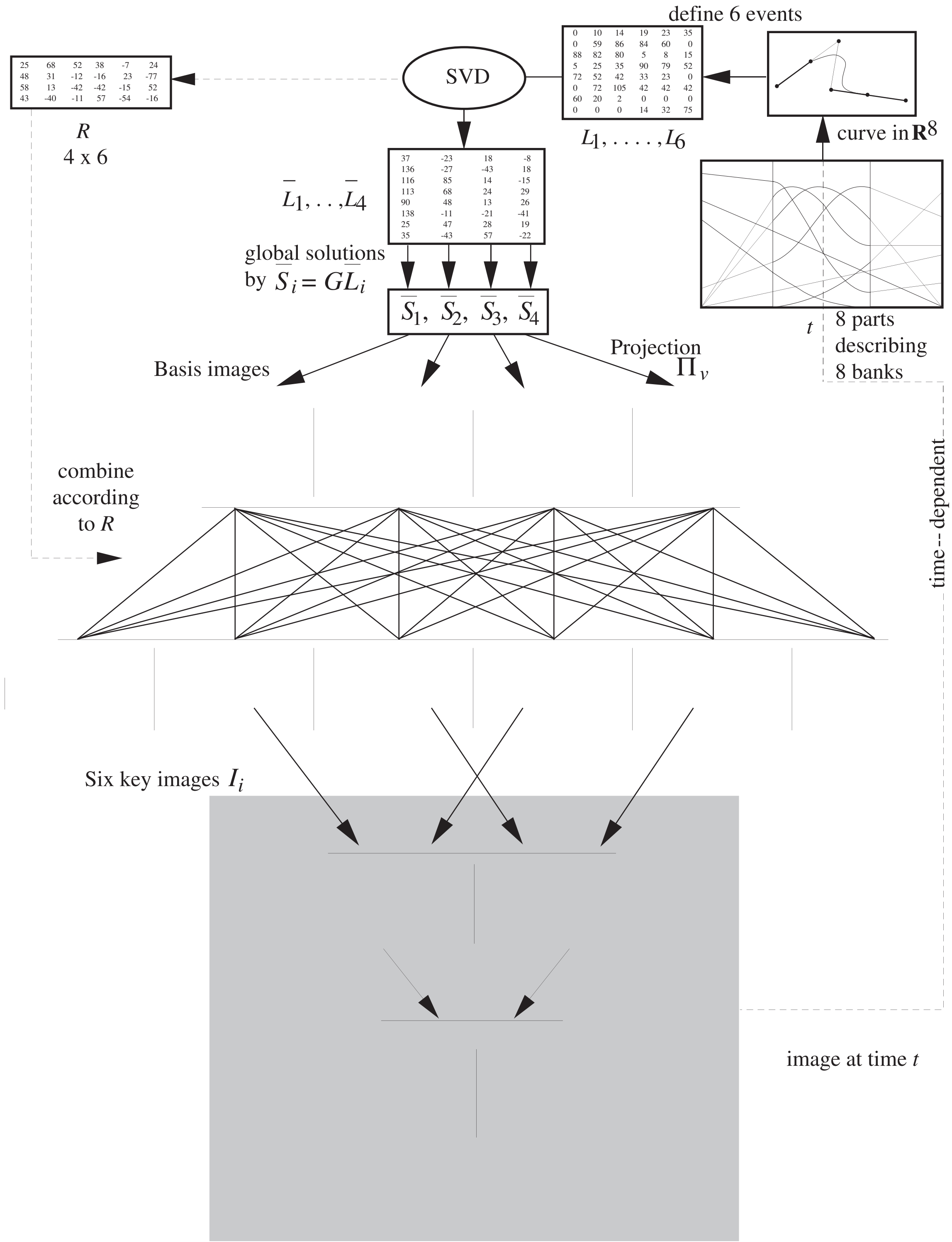


**Figure 4:** Flow chart of the full formulation with a real example.

sequence, we introduce incremental updates that can more quickly follow the path the designer takes as he/she explores the space of possible designs.

The obvious drawback to the use of SVD described in Section 4.1 is that it requires a complete specification of the parts before the optimal set of basis images can be determined and, therefore, before the first image is created. 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 more closely matching the needs of the design process, and by adding a bit more capability to the **G** operator.

At all times the designer must be able to quickly explore a wide variety of design alternatives. For this reason it would be ideal to have a separate image per bank before the design process even begins, as described in Section 2.2. 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.

### 5.1 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** will depend on the type of rendering algorithm it encapsulates. For example, a progressive radiosity approach can be tuned 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 anti-aliasing. 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 will refer to the former images as *coarse* and the latter as *refined*. This added control will allow us to construct a design tool that is amenable to progressive refinement and remains interactive at all stages of the design process.

### 5.2 Fast Feedback for Interactive Design

Initially a coarse image is computed for each of the $n$ banks of variable lights, as well as one for the single fixed bank. Once this is done, the feedback in the initial phase of the design process is very fast; when new key images are required, they can be quickly constructed 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 Section 2.2.

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.

### 5.3 Improving Preview Quality

As the design approaches its objective, more detailed feedback may be necessary than is afforded by the coarse images. Higher accuracy may be useful in capturing 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; for instance, 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. It is possible to get a greater 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 will provide the greatest improvement. Additional images can be computed in order of decreasing singular values, although the improvements will be correspondingly smaller.

**Figure 5:** Screen dump of the interactive lighting design program.

By viewing the sequence after each new image is added, the designer can evaluate the design at a number of intermediate points. However, it is reasonable to expect that changes will be made to the design as each improvement increases the quality of the playback. 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.

### 5.4 New Designs from Old Images

If the design is changed after some refined images have been computed, the refined images will still be of use, even though they do not match the new basis images suggested by SVD. Their utility is easy to see by imagining the $k$ refined images as the basis of a $k$-dimensional subspace of $\mathcal{R}^n$. This subspace can be extended to all of $\mathcal{R}^n$ 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 $k \leq n$ 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, as 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 is given 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 [6] is a simple means of computing this approximation and also for filling in the orthogonal component using the coarse solutions. By subtracting off the projection of the key setting vector on each element of the pool in descending order of quality (i.e. refined first, followed by coarse), the Gram-Schmidt procedure automatically makes maximal use of the refined images in approximating any given image.

### 5.5 SVD and the Orthogonal Component

In section 5.3 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 account of refined images that may have been previously computed. 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 insures that each new member is chosen to be of maximum benefit to the current design, never re-computing components that can be obtained through linear combinations of existing refined images.

### 5.6 Summary of the Design Methodology

The design methodology presented in this section fosters interactive design, as it gives the designer the ability to get fast feedback as a design is refined. It does so by using coarse approximations early on, by progressively refining the previews, and by taking full advantage of all refined images that have already been 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 $\alpha$-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 $\mathbf{G}$ 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.

## 6 Results

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

Figure 6 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; this is easily handled with multiple lights of different colors. Another example of changing color is the illumination from the windows.

Figure 7 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, which is 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. There is a slight drop in quality 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 re-used 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 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, one refined image per bank was computed

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

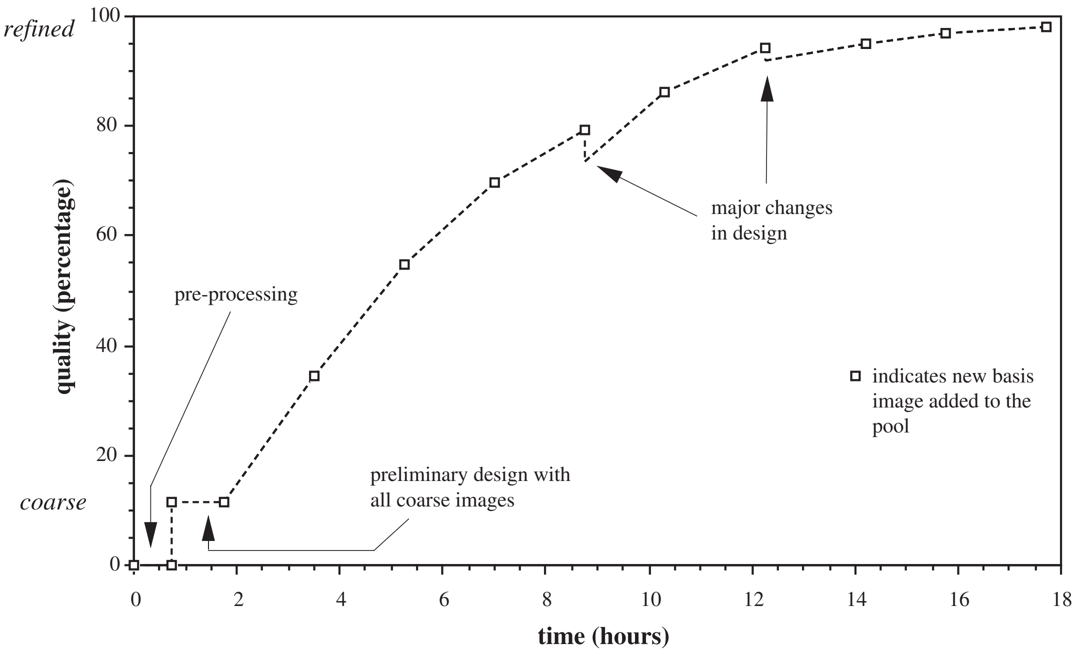


**Figure 7:** Quality versus time for a design session using the tavern scene. Timings were made on an HP 9000/750 workstation.

prior to the design session. The overall quality was then determined 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 one 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 is transformed 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 [4] 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.

**Figure 8:** Time-dependent simulation for *Turandot*.

## 7 Conclusion and Future Work

A set of tools for the interactive design and preview of complex, time-dependent lighting intensity variations is described. By constraining the problem to deal only with intensity variations, the formulation can fully exploit the linearity of light transport. Rapid playbacks are accomplished using at most three $\alpha$-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 as he/she explores the space of possible designs.

There are several future directions that we would like to pursue. In the current implementation we make use of two extreme quality levels when computing global illumination. We would like to generalize this to a continuum, although this 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 non-linear problems such as varying the geometry and material attributes.

## 8 Acknowledgements

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 Corporation generously provided the workstations used for the simulations. The Program of Computer Graphics 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.

## References


[1] Airey, J. M., J. H. Rohlf, and F. P. Brooks. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. *Computer Graphics*, 24(2):41–50, March 1990.

[2] Buckalew, C. and D. Fussell. Illumination Networks: Fast Realistic Rendering with General Reflectance Functions. *Computer Graphics*, 23(3):89–98, July 1989.

[3] Chen, S. E. Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System. *Computer Graphics*, 24(4):135–144, Aug. 1990.

[4] Dorsey, J. O., F. X. Sillion, and D. P. Greenberg. Design and Simulation of Opera Lighting and Projection Effects. *Computer Graphics*, 25(4):41–50, Aug. 1991.

[5] George, D. W., F. X. Sillion, and D. P. Greenberg. Radiosity Redistribution for Dynamic Environments. *IEEE Computer Graphics and Applications*, 10(4):26–34, July 1990.

[6] Golub, G. H. and C. F. Van Loan. *Matrix Computations*. Johns Hopkins, Baltimore, MD, 1983.

[7] Kajiya, J. T. The Rendering Equation. *Computer Graphics*, 20(4):143–150, Aug. 1986.

[8] Nimeroff, J., E. Simoncelli, and J. Dorsey. Efficient Re-rendering of Naturally Illuminated Environments. In *Proceedings of the Fifth Eurographics Workshop on Rendering (Darmstadt, Germany, June 13–15, 1994)*, pages 359–374, June 1994.

[9] Puech, C., F. Sillion, and C. Vedel. Improving Interaction with Radiosity-based Lighting Simulation Programs. *Computer Graphics*, 24(2):51–57, March 1990.

[10] Schoeneman, C., J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with Light. *Computer Graphics*, 27(2):143–146, Aug. 1993.

[11] Séquin, C. H. and E. K. Smyrl. Parameterized Ray Tracing. *Computer Graphics*, 23(3):307–314, July 1989.

[12] Ward, G. Real Pixels. In Arvo, J. R., editor, *Graphics Gems II*, Academic Press, New York, 1991.