Light Collages: Lighting Design for Effective Visualization

Chang Ha Lee Xuejun Hao Amitabh Varshney

Department of Computer Science and UMIACS
University of Maryland at College Park
College Park, MD 20742
{chlee, hao, varshney}@cs.umd.edu

ABSTRACT

We introduce Light Collages – a lighting design system for effective visualization based on principles of human perception. Artists and illustrators enhance perception of features with lighting that is locally consistent and globally inconsistent. Inspired by these techniques, we design the placement of light sources to convey a greater sense of realism and better perception of shape with globally *inconsistent* lighting. Our algorithm segments the objects into local surface patches and uses a number of perceptual heuristics, such as highlights, shadows, and silhouettes, to enhance the perception of shape. We show our results on scientific and sculptured datasets.

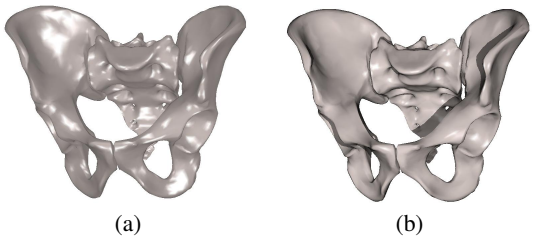
CR Categories: I.3.3 [Computing Methodologies]: Computer Graphics—Picture/Image Generation; I.3.8 [Computing Methodologies]: Computer Graphics—Applications

Keywords: Lighting design, scientific illustration, inconsistent lighting, light placement, silhouette enhancement, proximity shadows

1 INTRODUCTION

The last two decades have witnessed impressive advances in algorithms for lighting simulation. However, effective lighting design has remained a challenge. Effective lighting design can convey a large number of data features such as local surface orientation, curvature, silhouettes, and fine texture. Current lighting design techniques include automatic placement of lights based on user-specification of shadows and highlights [19, 20]. Inverse lighting models have proven to be powerful in lighting design [3, 4, 7, 21, 22, 29]. Such models allow one to compute the intensities of light sources from photographs, or in general, user-specified light distributions. Recent advances in relighting are based on linear superposition of light transport. Thus, one method for lighting design involves compositing multiple images under varying lighting. Akers *et al.* [1] have shown how image composition can be used with sophisticated, spatially-varying light mattes to create compelling technical illustrations from a set of photographs of an object.

Our interest in lighting design for effective scientific visualization is inspired by research that examines the human perception of art. Over two millennia ago Pliny the Elder described the technique of locally shading a surface fold to make it appear to rise above the background [8]. Since then, similar local techniques for lighting have been successfully used by artists and illustrators to convey lighting and shading. What is interesting about these techniques is that they convey a powerful impression of geometry, although the lighting across the surface is inconsistent. It is even more interesting that although the geometry of perspective has been well understood for the past five centuries, the geometry of consistent lighting has been largely ignored in art. Thus, if one were to apply the inverse lighting models that have been developed recently [3, 21, 29] to most paintings and illustrations, one would find innumerable errors (some admittedly slight, but present nonetheless) in their lighting and shading. However, not only have these lighting errors passed virtually unnoticed by most untrained human observers, lighting for such paintings is visually impressive and sometimes even deeply compelling.



(a) (b)

Figure 1: *(a) Consistent lighting with four lights at the vertices of a regular tetrahedron, and (b) a Light Collage rendering with 4 lights. Material properties are the same for both renderings.*

Artists use inconsistent lighting for a couple of reasons. First, it is convenient. Consistent lighting is often not worth the effort since hardly anyone notices it. Second, the artists can use inconsistent lighting to guide the viewer's attention and enhance comprehensibility. Cavanagh [5] has suggested that our brain perceives the shape-from-shading cues locally and does not use large regions of the visual field for shape-from-shading analysis. In this paper we explore the implications of globally inconsistent lighting for enhancing the visualization of scientific datasets. Inconsistent lighting, as used in art, might allow us to convey a better perception of geometry than consistent lighting. Here we use this observation in optimizing light placement to convey a greater sense of realism when visualizing scientific datasets.

The main contributions of this paper are:

- *Globally Inconsistent Lighting:* This paper discusses a method of local illumination that is locally accurate, globally inconsistent, and yet presents more comprehensible renderings.
- *Lighting Design:* We discuss placement of multiple light sources to enhance view-dependent visualization.
- *Feature Enhancement:* Silhouettes and shadows add important details in technical illustrations. Here we show how silhouette and shadow lighting can be integrated into a local illumination framework for enhancing features in scientific datasets.

2 PREVIOUS AND RELATED WORK

Lighting design has long been considered crucial in photography, cinematic lighting and stage lighting. Lighting design for computer

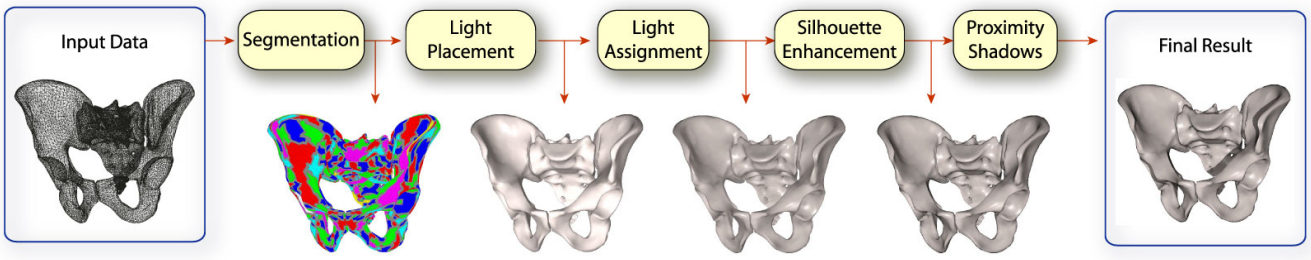


Figure 2: *Overview of our lighting design pipeline: The input model is segmented using a curvature-based-watershed method into a set of patches. The light placement function models the appropriateness of light directions for illuminating the model. This is done by using the curvature-based segmentation as well as the diffuse and specular illumination at every vertex. Lights are placed and assigned to patches based on the light placement function. Silhouette lighting and proximity shadows are added for feature enhancement.*

graphics involves specifying the parameters for light position, direction, color, intensity, and the illumination model. Traditional lighting design methods for graphics have been *direct* – they have required a user to directly specify the lighting parameters. While direct light specification is satisfactory, it often requires significant expertise on the part of the user to adequately communicate the visual features of a graphics scene. Direct lighting design is often iterative and time consuming. The user starts out by specifying an initial set of lighting parameters and then visually evaluates the results. The lighting parameters are then changed iteratively till the graphics rendering converges to a desired output. This is often a tedious process. The approach of Design Galleries [17] addresses this by using several user-specified lighting parameters (excluding light placement), generating a set of renderings with randomly placed lights, and having a user browse and linearly combine the renderings that are desirable. The LightKit system [11] allows a user to interactively adjust lighting to enhance visualization. This system allows camera-relative lights that include a dominant light, headlights, and backlights. The system also allows the user to adjust the light color and warmth of lighting.

Indirect lighting design methods use scene properties that are either specified by a user or procedurally estimated. In *user-specified indirect lighting design*, the user specifies the desired highlights or shadows and the system then infers the light placement to achieve them [6, 15, 19, 20, 23]. In *procedural indirect lighting design*, the system automatically infers light placement and parameters by optimizing a set of perceptual criteria for a given view. Shacked and Lischinski [24] derive light placement for up to two light sources by optimizing a perception-based image quality objective function. Their objective function is comprehensive and includes six terms that are based on shading gradients, pixel luminance statistics, and illumination direction. Gumhold [10] has developed a light-placement strategy by maximizing the illumination entropy as measured from a rendered image. Based on user studies he has then developed a perceptual entropy objective function that incorporates a surface-curvature-based importance weighting.

Non-Photorealistic lighting differs from the above in that it does not restrict itself to physically correct lighting and shading. Gooch *et al.* [9] have developed a lighting model that uses luminance and changes in hue to convey surface orientation, edges, and highlights. Hamel [12] has developed a lighting model that incorporates five components – standard lighting with shadows, rim shadow lighting, curvature shading, transparency, and volume illumination. Sloan *et al.* [25] have developed an effective method to transfer the shading from one object to another using a sphere (environment map) as an intermediary. Anderson and Levoy [2] have used curvature- and accessibility-based shading [18] to enhance the visualization of cuneiform tablets. Vicinity lighting [26] improves upon the idea of accessibility shading by using uniform diffuse lighting and occlusion by local occluders.

Cinematography is a major application of lighting design. Kahrs *et al.* [13] have summarized the lighting design approaches for computer animation. *Logical* lights are motivated by actual sources of light in a scene that the viewer can see or imply. For example, the *key* light is used in a scene as the primary source of illumination. In addition to logical lighting, cinematographers also use *pictorial* lighting, just for enhancing the artistic aesthetics of the scene. For example, *fill* lights are used to soften and fill the shadows, and *back or rim* lights are used to separate the object from the background.

To the best of our knowledge, none of the previous work has tried to 3D render the same object with multiple light sources with each light source lighting a different region of the object. In fact, the general advice seems to have been to illuminate objects with a single light source that is placed above and to the left of the object [27]. In this paper we introduce the idea of *Light Collages* that involves lighting different regions of a 3D object with multiple light sources to render it in a more visually comprehensible manner, while retaining its traditional 3D-graphics-rendered look and feel.

3 Light Collages

The goal of our Light Collages framework is to assign local lights to different regions of an object. Specifically, consider an object composed of n surface patches $\mathscr{P} = \{p_1, p_2, \ldots, p_n\}$. Let there be a set of m unknown light sources $\mathscr{L} = \{l_1, l_2, \ldots, l_m\}$. The problem we solve here is: *Given $\mathscr{P}$, m, and a viewer position, generate $\mathscr{L}$ and a mapping $\mathscr{M}$ that pairs each light $l_j \in \mathscr{L}, 1 \leq j \leq m$ to a subset of patches $\mathscr{P}_i \subset \mathscr{P}$ that it lights, to best elucidate the local structure of the object*. Each patch p_i is assigned a primary light source $l_j = \mathscr{M}(p_i)$. Here, obviously, what constitutes the *best elucidation* is open to interpretation. We assume that conveying the local curvature information is important. In fact user studies on light source placement by Gumhold [10] have indicated that observers tend to select light source directions that favor surface curvature elucidation. In this paper, we only consider directional light sources.

3.1 Surface-Curvature-based Segmentation

We use a surface-curvature-based segmentation of an object to better elucidate the object shape. The segmentation of an object is a classical area of research in computer vision and image processing. Recent work in computer graphics discusses how polygonal meshes may be segmented to yield visually intuitive segmentation [14]. Any of the vast number of segmentation algorithms can be used for object segmentation at this stage depending on what the goals of the segmentation-based lighting design are. In this paper, we segment the object into patches based on local curvature. For this, we first compute the mean curvature at each vertex of the input mesh as the average of its two principal curvatures, which are computed using

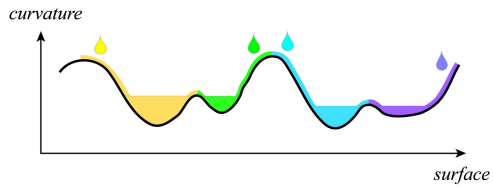
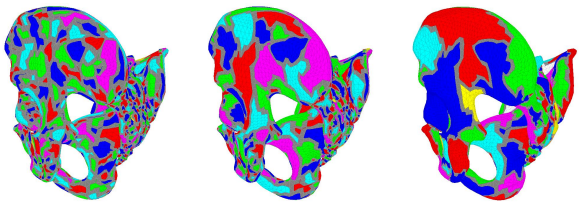


Figure 3: *The watershed algorithm: First, we assign unique labels (patch IDs) to local minimums. Next, imagine that we place a drop of water at a vertex. The water drop will flow to the local minimum. We assign the label of the local minimum to the vertex where the water-drop was placed. We repeat this for each vertex.*

Taubin's method [28]. We then use a simple watershed algorithm based on Mangan and Whitaker's method [16].

Mangan and Whitaker's method segments a mesh into patches such that each patch has a largely similar curvature and is surrounded by regions of drastically different curvature. Their method proceeds by finding vertices with local curvature minima and using these as seeds for growing new patches. The method then iteratively assigns vertices to these patches. A path of steepest descent is computed from each unassigned vertex till it reaches a seed vertex with a local curvature minimum. The vertex is assigned to the patch corresponding to this seed vertex. A *watershed depth* is computed for each patch based on the minimum difference in curvature values between a boundary vertex and the seed vertex for that patch. Since this phase of the algorithm results in over-segmentation, it is followed by a patch-merging phase in which patches whose watershed depth is below a threshold depth are merged. This is shown in Figure 3. The segmentation can be increased or decreased by respectively lowering or raising the threshold depth. This is shown in Figure 4. Figure 5(a) shows the distribution of the curvature over the Pelvis model and Figure 5(b) shows the results of our segmentation of the object into multiple surface patches: $\mathscr{P} = \{p_1, p_2, \ldots, p_n\}$. Each patch is a collection of triangles with similar curvature values.

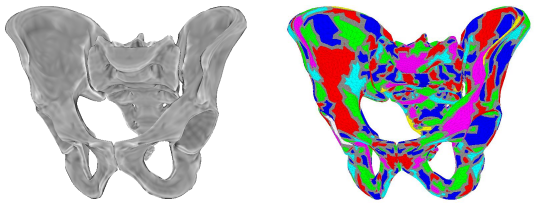


(a) (b) (c)

Figure 4: *Segmentation of Pelvis model at different thresholds: For all models in this paper, we use 7.5% of the range of curvature difference as a threshold. Figures (a), (b), and (c) show a low level (3.0%), a middle level (7.5%), and a high level (25%) segmentation, respectively.*

3.2 Light Blending and Normalization

As discussed earlier, our Light Collages framework illuminates local surface patches with globally inconsistent lighting. A straightforward implementation of this idea results in sharp visual discontinuities across patch boundaries that are lighted differently. Such shading discontinuities are disconcerting especially when they occur in absence of shape discontinuities. To alleviate such visual artifacts we blend illumination from neighboring patches. As mentioned earlier, every vertex i in a patch p_j is illuminated by light $\mathscr{M}(p_j)$. The illumination at a vertex i is a weighted sum of illuminations from the primary lights for all the patches $\mathscr{N}_j$ that neighbor patch p_j: $\mathscr{N}_j = \{p_k \mid \partial p_j \cap \partial p_k \neq \emptyset\}$, where ∂p_j denotes the



(a) (b)

Figure 5: *In (a) we show curvature distribution on the surface. Convex regions are shown brighter and concave regions are darker. Figure (b) shows the results of our curvature-based segmentation.*

boundary of patch p_j. Let the primary light for patch $p_k \in N_j$ be given by $l_k = \mathscr{M}(p_k)$. Let the weight of vertex i with respect to the primary light of patch p_k be based on the distance function $d()$ of vertex i from the boundary ∂p_k and be given by:

$$w_{ik} \propto \frac{1}{1 + d(i, \partial p_k)}$$

We note that the distance $d(i, \partial p_j)$ is zero for a vertex inside or on the boundary of the patch p_j. Therefore, the weight of vertex i in patch p_j, $w_{ij} = 1$.

Let the illumination at vertex i due to light l_k be given by $I_i(l_k)$. Then the total illumination at vertex i in patch p_j will be given by $\overline{I_i} = \sum_k w_{ik} I_i(l_k)$, where $p_k \in \mathscr{N}_j$. The distribution of the blending weights at vertices around a patch is shown in Figure 6.

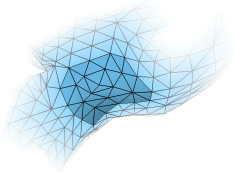


Figure 6: *Visualization of weights for blending illumination from a patch. The dark area in the middle shows the patch, and the progressively whiter colors outside the patch show the weight values diminishing away from the patch boundary.*

The overall brightness of a rendered image is perceptually important. It is desirable to avoid rendering an object with too much brightness since it tends to diminish the discriminability amongst object features. To achieve a balanced rendering brightness we normalize the illumination as computed above with the blending weights for a given vertex. Thus, the final illumination formula for a vertex i in patch p_j with neighbors $\mathscr{N}_j$ is given by:

$$\overline{I_i} = \frac{\sum_k w_{ik} I_i(l_k)}{\sum_k w_{ik}}, p_k \in \mathscr{N}_j, l_k = \mathscr{M}(p_k)$$

4 Procedural Lighting Design

Lighting design is a very broad area that involves specification of a number of lighting parameters such as number and type of lights, their placement, and their colors. Here we focus on light placement and assignment of lights to surfaces. We assume that all the lights are white and directional. Our lighting design proceeds in two interleaved phases. In one phase we identify the placement of a light and in the other phase we assign the light to appropriate patches.

4.1 Light Placement

Humans use several visual cues to comprehend shapes and distinguish among objects. One of the important features of shapes is their curvature. Curvature influences the illumination gradient across the surface. We use a combination of local lighting models to enhance the appearance of high-curvature areas of an object from a given viewpoint. The reflectance of an object consists of three components: ambient, diffuse, and specular. The ambient color does not vary across an object. The diffuse color varies according to the Lambertian law. The illumination component that changes the fastest is specular. A specular highlight on a shiny surface can easily vanish with even small perturbations of the viewing direction, surface normal, or light direction. For an object with high curvature, the specular highlight is useful as it can result in a sharp curvature-based highlight, and thus help illustrate object detail.

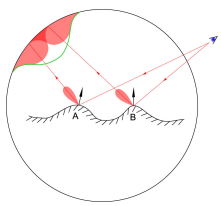


Figure 7: *Finding good light directions: For a given viewing direction, a weight function for each surface point is added to the light placement function defined in the directional space (shown here by the large circle). The value of the light placement function in a direction is related to the appropriateness of placing a light along that direction.*

Let us consider a simple example with two points A and B on which we would like to place specular highlights (Figure 7). If we have the freedom to place a directional light source along any direction, we would like to place that light source in a direction that maximizes the possibility of having highlights on points A and B. By using the view direction, the shape, and the material properties of points A and B, we can infer the light directions that will cause specular highlights to appear on them. Using the reciprocity principal, this is equivalent to shooting a ray of light from the viewpoint to the points A and B, and having that light specularly reflect out to the environment. The specularly reflected rays will result in a distribution around the direction of mirror reflection. This is shown in Figure 7. The red blobs on the upper left region of the circle represent the probability density function (PDF) of the reflected ray along those directions. The total probability of a specular highlight can be computed by the sum of the individual PDFs, as shown by the green curve. Thus, following the reciprocity principal, if we were to place a light source in the direction where the green curve has the largest value, we would get the best highlights at both the points A and B for the given view position.

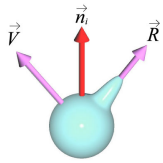


Figure 8: *Computation of specular weight function for a vertex with normal $\vec{n}_i$ and curvature intensity c_i: Let $\vec{R}$ be the reflection of viewing direction $\vec{V}$ at vertex i. This means that the light placed along $\vec{R}$ maximizes the specular illumination at vertex i. Specular weight function $S(i,\vec{l})$ is defined as the fall-off function around $\vec{R}$ weighted by curvature.*

We extend the above ideas to define a light placement function $P(\overrightarrow{l})$ that models the appropriateness of placing a light in the direction $\overrightarrow{l}$. Such a light placement function should include contributions from both specular as well as diffuse illumination. We next discuss how we compute the specular- and the diffuse-illumination-based factors that will go towards defining the overall light placement function.

As before, let $\mathscr{P}$ be the set of surface patches for an object. Let $\overrightarrow{v}$ be the view vector, $\overrightarrow{l}$ be the light direction, and $\overrightarrow{h}$ be the halfway unit vector along the direction $\overrightarrow{l}+\overrightarrow{v}$. Further, let κ_i be the mean curvature and $\overrightarrow{n}_i$ be the normal vector at a vertex i on the surface. We define the specular weight function S for the vertex i with the light along $\overrightarrow{l}$ with a shininess s as:

$$S(i,\overrightarrow{l}) = |\,\kappa_i\,|\,(\overrightarrow{n}_i \cdot \overrightarrow{h})^s$$

Given a view direction, we compute $S(i,\overrightarrow{l})$ for each vertex i and for a set of uniformly distributed light directions $\overrightarrow{l}$. In our implementation we use $12K$ uniformly distributed directions $\overrightarrow{l}$. The use of specular highlights alone is not desirable, as shown by Gumhold [10]. In addition to specular lighting, we would also like to consider diffuse lighting when computing the light placement function to guide us in light source placement. Since observers have found curvature to be informative in lighting design, we have designed our diffuse lighting component to adapt to the local curvature on a patch-by-patch basis. Figure 5(a) shows a visualization of curvature distribution. We define the *curvature intensity* c_i at a vertex i to be its normalized mean curvature, i.e. $c_i = (\kappa_i - \kappa_{min})/(\kappa_{max} - \kappa_{min})$, where κ_i is the mean curvature at vertex i, and κ_{max} and κ_{min} are the maximum and minimum values of the mean curvature among all the vertices of the input mesh, respectively. For a vertex i with normal vector $\overrightarrow{n}_i$, let $\mathscr{D}$ be the set of light directions whose diffuse color is same as the curvature intensity c_i.

$$\mathscr{D} = \{\overrightarrow{d}\,|\,\overrightarrow{d}\cdot\overrightarrow{n}_i = c_i\}$$

We define the diffuse weight function $D(i,\overrightarrow{l})$ for vertex i in the direction of $\overrightarrow{l}$ such that the diffuse illumination at vertex i is similar to the curvature intensity c_i. We compute it as the upper envelope (maximum) of the dot product between $\overrightarrow{l}$ and all $\overrightarrow{d} \in \mathscr{D}$:

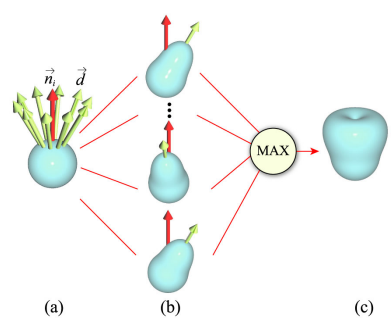


Figure 9: *Computation of diffuse weight function for a vertex with normal $\vec{n}_i$ and curvature intensity c_i: First, (a) we define the set of light directions $\vec{d} \in \mathscr{D}$ for which $\vec{n}_i \cdot \vec{d} = c_i$. These directions $\vec{d}$ are shown by green arrows. If we were to place a light along any $\vec{d} \in \mathscr{D}$, the diffuse intensity at this vertex $\vec{n}_i \cdot \vec{d}$ will be proportional to c_i. Figure (b) shows the cosine fall off about each direction $\vec{d} \in \mathscr{D}$, computed as the dot product of an arbitrary vector $\vec{l}$ with $\vec{d}$, for all $\vec{l}$. (c) The diffuse weight function $D(i,\vec{l})$ is the upper envelope (maximum) of the functions shown in Figure (b).*

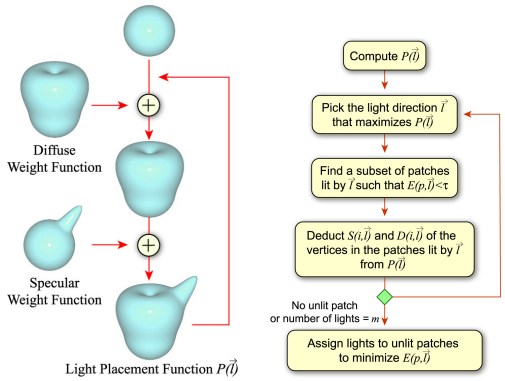


Figure 10: *The light placement function $P(\vec{l})$ is computed in Figure (a) by adding diffuse and specular weight functions. Figure (b) shows the flowchart of the process for light placement and assignment.*

$$D(i,\overrightarrow{l}) = \underset{\overrightarrow{d}\in\mathscr{D}}{Max}\ \overrightarrow{l}\cdot\overrightarrow{d}$$

The light placement function can be computed as the sum of specular and diffuse weight functions over all surface points. For any light direction $\overrightarrow{l}$ the value of the light placement function $P(\overrightarrow{l})$ along that direction is given by:

$$P(\overrightarrow{l}) = \sum_i (S(i,\overrightarrow{l}) + D(i,\overrightarrow{l}))$$

4.2 Assignment of Lights to Patches

We use the light placement function $P(\overrightarrow{l})$ to select the best m lights $\mathscr{L} = \{l_1, l_2, ..., l_m\}$, as follows. We identify the light direction $\overrightarrow{l}$ that maximizes $P(\overrightarrow{l})$. We select this to be the direction of the first light l_1. We then identify the patches which will be lit by the light l_1. For any light $l_k \in \mathscr{L}$ and patch $p \in \mathscr{P}$, let $\mathscr{S}_p$ be the set of points that are on p. We define a function $E(p,l_k)$ that measures the similarity of the illuminated intensity I_i for vertices i in the patch p (as defined in Section 3.2) to its curvature intensity as:

$$E(p,l_k) = \sum_{i\in\mathscr{S}_p} (I_i(l_k) - c_i)^2$$

So, for the first light l_1, we compute $E(p,l_1)$ for every $p \in \mathscr{P}$, and if $E(p,l_1)$ is less than a threshold τ (currently we use $\tau = 0.15$), l_1 is assigned to p, i.e. $\mathscr{M}(p) = l_1$. We deduct the contributions of the vertices in the patches lit by this light l_1 from the light placement function. Then we again identify the light direction that maximizes the updated light placement function, and select that to be the direction of the second light source l_2. We repeat this process until m lights are selected. Figure 11 shows the lighting with one, two, and four lights. Patches that are not lighted are shown dark without any blending with the neighboring patches.

After we have chosen m light sources, most of the patches will be lit by some light source. However, some patches may remain unlit. For each unlit patch, one of the m light sources is assigned to it using the function $E(p,l_k)$ defined in the previous section. For each patch p, $E(p,l_k)$ is computed for all m lights, and the light l_k which minimizes $E(p,l_k)$ is assigned to p.

$$\mathscr{M}(p_i) = \underset{l_k\in\mathscr{L}}{Argmin} E(p,l_k)$$

Figure 12 shows the light directions for some of the patches.

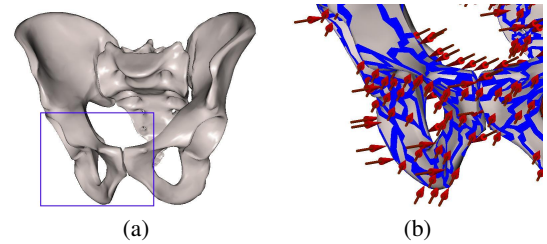


Figure 12: *(a) Rendering of the pelvis model in which each surface patch is lighted by one out of 8 directional light sources. Figure (b) is enlarged view of the region selected by the rectangle in Figure (a) and shows the primary light directions for some of the patches. The blue stripes show the boundaries between patches.*

5 Feature Enhancement

5.1 Silhouette Lighting

Silhouettes are important for defining the boundary of objects. A well-defined silhouette makes an object easier to comprehend and makes its shape more distinguishable. In cinematography, backlights are used for separation of foreground from background. Backlights are traditionally placed behind the object generating a thin rim of light around the silhouette of the object. Backlights are also referred as rim, hair, or separation lights. In particular, the lights at the three-quarters-back position are called as kicker lights [13].

To make the objects stand out from their background, we would like to produce a dark silhouette for a bright background and a bright silhouette for a dark background. To enhance the contrast between a silhouette and its background, we use a simple fall-off formula weighted by $\omega_s = (1 - \overrightarrow{n}_i \cdot \overrightarrow{v})^u$, for adding an additional silhouette light at vertex i with normal $\overrightarrow{n}_i$ and view direction $\overrightarrow{v}$. The results of incorporating black silhouette lighting appear in Figure 13. We compute the silhouette-enchanced illumination as the linear blend of the silhouette lighting H_i weighted by ω_s and the existing illumination: $(1-\omega_s)I_i + \omega_s H_i$.

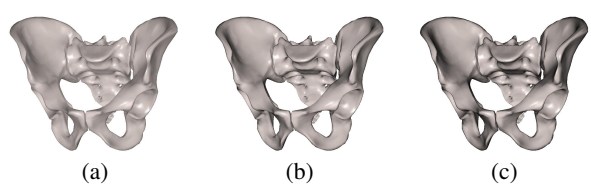
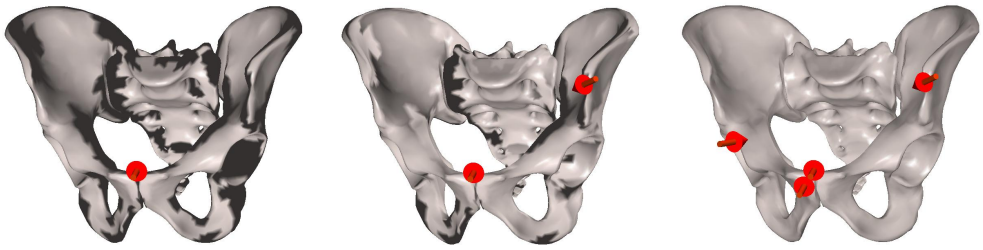


Figure 13: *Light Collages rendering (a) without, and (b), (c) with silhouette lighting. In this figure, $(1-\vec{n}_i\cdot\vec{v})^u$ is used as the silhouette light's weight factor. The silhouette lighting H_i is black to make near-silhouette regions dark since the background is bright. (b) and (c) show different levels of silhouette enhancement with $u = 4$ and $u = 2$ respectively.*

5.2 Proximity Shadows

Shadows present a very important visual cue. Carefully placed shadows can greatly enhance features and make spatial relationships between regions clearer. As an example, it may be difficult to distinguish two surface patches if they have similar illumination but different distances from the viewer and overlap in space as seen by the viewer. If however, the front patch casts a visible shadow on the back patch, then the viewer will have no ambiguity in recognizing their spatial relationship. Two visible patches that overlap in a given view, but are at different distances from the viewer will result in a depth discontinuity in the computed depth map. The depth



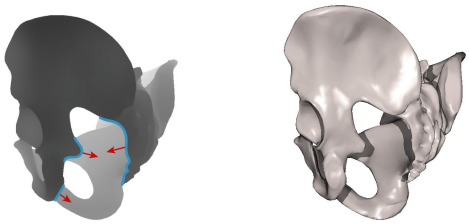
(a) (b) (c)

Figure 11: *Partial surface lighting with (a) first light, and (b) first two lights, and (c) four lights. The red arrows show the light directions. The dark regions in (a) and (b) are the patches which are not lit by current partial lights. No blending is used here.*

discontinuity usually occurs along one or more curves as shown in Figure 14 (a). We use proximity shadows to show the relative distances between the two overlapping patches if their depths are within a predefined threshold value.

First, we identify the depth discontinuity curves by comparing the value of each pixel in the depth map with its neighbors. We then generate a shadow light direction for each depth discontinuity curve by using the depth gradient. We compute the per-pixel depth gradient by using the central differences method that examines the depth variations in the immediate vicinity of a pixel. We then average the per-pixel depth gradient over all the pixels of a discontinuity curve to arrive at an average value of the depth gradient for the curve. The shadow light direction is determined by rotating the viewing direction by a small angle θ towards the average depth gradient of the depth discontinuity curve as shown in Figure 15. Finally we use the shadow light direction in a shadow map to cast proximity shadow for the depth discontinuity curve. This process is repeated for all the depth discontinuity curves.

During the generation of local proximity shadows, we have to be aware of one possible problem – a narrow region might have depth discontinuity on both of its sides. If we cast shadows of this region on both surfaces that are behind, it can produce a somewhat disconcerting effect as shown in Figure 17(b). For such situations one can use any heuristic that consistently picks one side of the region over the other. Examples of such heuristics may include picking the side of the discontinuity region that is on the left and the top, or pick the side of the discontinuity region that has more surface points on the discontinuity curve and use it to cast shadows as shown in Figure 17(c).



(a) (b)

Figure 14: *(a) A set of adjacent points whose depth differences with neighbors are greater than a threshold forms a discontinuity curve in the depth map. The arrows show the average gradients of discontinuity curves. Figure (b) shows the proximity shadows casted by the discontinuity curves in (a).*

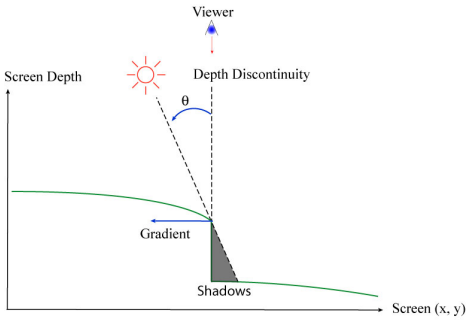
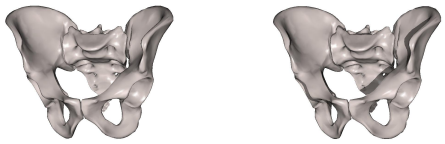


Figure 15: *The placement of a light for proximity shadow: At each depth discontinuity curve of the depth map, a light for the proximity shadow is placed by rotating a vector to the viewer at an angle θ along the direction of the local gradient.*



(a) (b)

Figure 16: *Rendering (a) without, and (b) with proximity shadows.*

6 Results and Conclusions

We have implemented the Light Collages system in OpenGL. The visualization results using our system can be seen in Figures 18 – 19. In Figure 18(a)–(c) we show the result of lighting the skull model by consistent lighting with 1, 2, and 4 lights, and (d)–(f) show the results by Light Collages rendering with 1, 2, and 4 lights. We have used the same lighting and material properties for generating all the six images. As you can see in (a)–(c), the specular highlight from consistent lighting will sometimes cause large bright areas on flat regions, while highlights from our methods ((d)–(f)) are only on highly curved regions. This helps elucidate geometry details. The proximity shadow cast by the upper cheek bone in Figures 18(d)–(f) nicely illustrates the depth relationship between these two regions of the skull. In Figure 19(a) we show the result of lighting the pelvis model lighted with the best 8 lights determined by our system. Figures 19(b) and (c) progressively add silhouette lighting and proximity shadows to the Figure 19(a).

We have introduced Light Collages as a system for automatic lighting design for effective visualization of scientific datasets. Our method relies on using multiple light sources that can be used for accurate local lighting on surfaces, with possible global inconsistencies. The human visual system is remarkably adept at inferring shape from largely local cues, and hence our system, even with global lighting inconsistencies can produce comprehensible renderings. We have shown how our method can incorporate silhouette

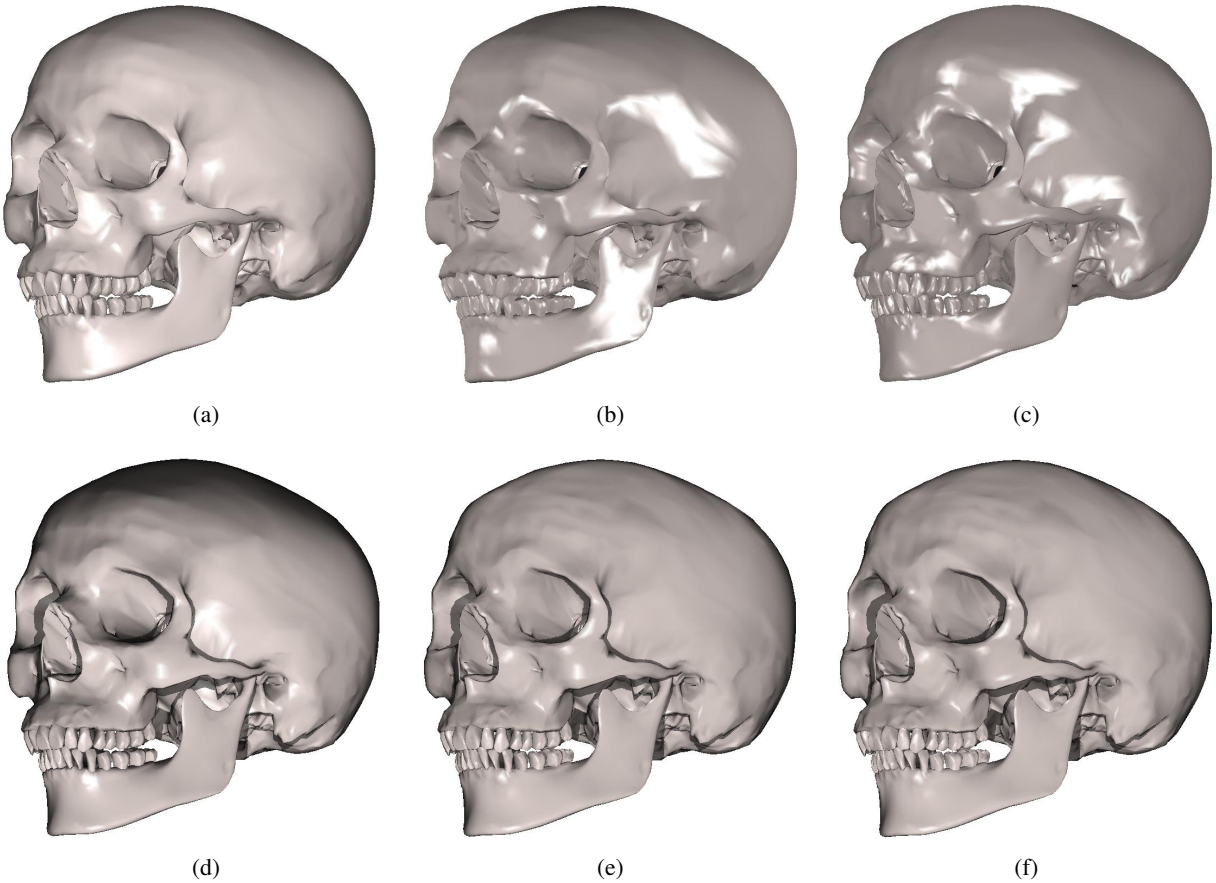


Figure 18: *Lighting for Skull: (a)–(c) use consistent lighting and (d)–(f) use Light Collages. (a) Lighting by one light at viewer, (b) Lighting by 2 lights at front vertices of a tetrahedron, (c) Lighting by 4 lights at front vertices of a cube. (d) Light Collages rendering using 1 light, (e) 2 lights, and (f) 4 lights.*

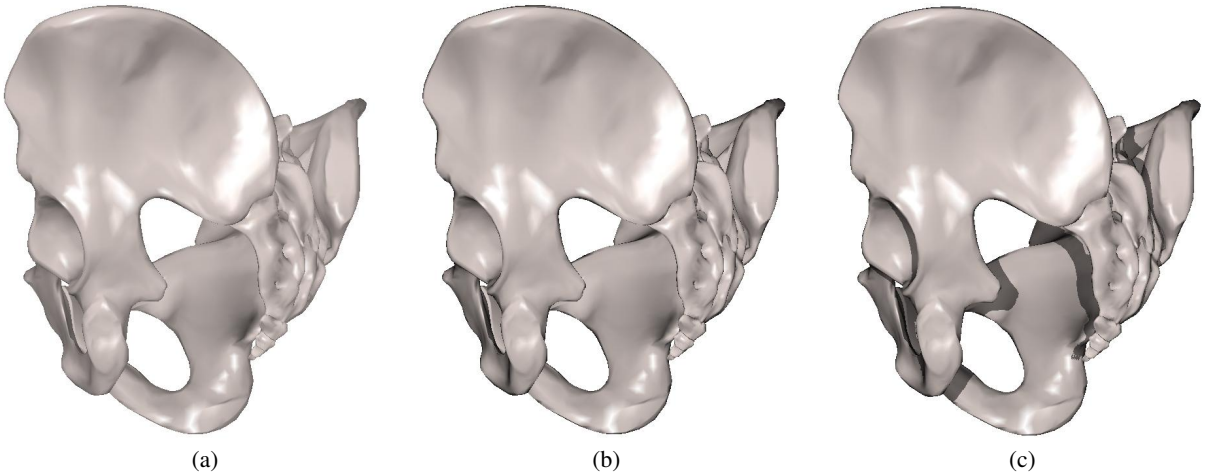
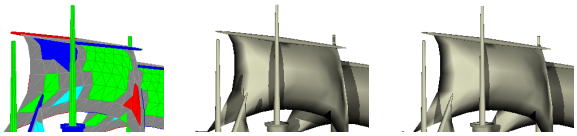


Figure 19: *Light Collages for Pelvis: (a) Lighting by 8 lights, (b) Adding silhouette lighting to the previous image, and (c) Adding proximity shadows to the previous image.*



(a) (b) (c)

Figure 17: *Avoiding Conflicts in Proximity Shadows: (a) Discontinuity curves are along the both sides of the mast. The patch for the mast is the common patch of the two depth-discontinuity curves. (b) The discontinuity curves at both sides of the patch result proximity shadows on both sides of the region. (c) Proximity shadows on both sides can be corrected by eliminating one of them.*

lighting as well as proximity shadows to further elucidate the local structure of the scientific datasets. We believe our method greatly improves the visualization while retaining the look and feel of traditional 3D graphics illumination models.

In addition to the visual appearance, interactivity is essential for the perception of 3D shapes. We have not yet worked to optimize the running times of our Light Collages system. We plan to use spherical-harmonics-based representations to efficiently compute the light placement function for use in an interactive system. In this paper we have assumed the lights are directional. Generalizing our approach to point light sources or perhaps even area light sources would be an interesting direction for future work. Our current work does not take into account variations in color or material properties and this should be useful to consider as well. In addition to lighting design for a single object, automatically designing lighting environments for a scene with multiple objects should be useful.

Acknowledgements

We would like to thank Martin Reddy, Tony DeRose, and Patrick Cavanagh for inspiring us to explore lighting design with multiple and inconsistent lights. David Jacobs, David Mount, and Youngmin Kim have helped in significantly improving the presentation of ideas in this paper. We would also like to acknowledge the anonymous referees for their exceptionally thorough reviews for this paper that have led to a much better presentation of our results. This work has been supported in part by the NSF grants: IIS 00-81847, CCF 04-29753, and CNS 04-03313.

References

[1] D. Akers, F. Losasso, J. Klingner, M. Agrawala, J. Rick, and P. Hanrahan. Conveying shape and features with image-based relighting. In *IEEE Visualization 2003*, pages 349 – 354, 2003.

[2] S. Anderson and M. Levoy. Unwrapping and visualizing cuneiform tablets. *IEEE Computer Graphics and Applications*, 22(6):82–88, November/December 2002.

[3] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, pages 383–390, July 9–12 2001.

[4] B. Cabral, N. Max, and R. Springmeyer. Bidirectional reflection functions from surface bump maps. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21(4), pages 273–281, July 1987.

[5] P. Cavanagh. Pictorial art and vision. *MIT Encyclopedia of the Cognitive Sciences*, pages 644–646, 1999.

[6] A. C. Costa, A. A. de Sousa, and F. N. Ferreira. Lighting design: A goal based approach using optimization. In *Rendering Techniques '99*, pages 317–328. Springer Wien, 1999.

[7] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In Kurt Akeley, editor, *SIGGRAPH 2000*, pages 145–156, 2000.

[8] E. H. Gombrich. *The Heritage of Appelles*. Oxford: Phaidon Press, 1976.

[9] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In Michael Cohen, editor, *SIGGRAPH 98*, pages 447–452, 1998.

[10] S. Gumhold. Maximum entropy light source placement. In Robert Moorhead, Markus Gross, and Kenneth I. Joy, editors, *IEEE Visualization 2002*, pages 275–282, 2002.

[11] M. Halle and J. Meng. Lightkit; a lighting system for effective visualization. In *IEEE Visualization 2003*, pages 363 – 370, 2003.

[12] J. Hamel. *A New Lighting Model for Computer Generated Line Drawings*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2000.

[13] J. Kahrs, S. Calahan, D. Carson, and S. Poster. Pixel cinematography: A lighting approach for computer graphics. In *SIGGRAPH Course Notes*, 1996.

[14] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, July 2003.

[15] J. K. Kawai, J. S. Painter, and M. F. Cohen. Radiooptimization - Goal Based Rendering. In *Proceedings of (ACM SIGGRAPH) '93*, pages 147–154, 1993.

[16] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[17] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. *SIGGRAPH 97*, 31:389–400, August 1997.

[18] G. Miller. Efficient algorithms for local and global accessibility shading. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 319–326, 1994.

[19] F. Pellacini, P. Tole, and D. P. Greenberg. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):537–546, 2002.

[20] P. Poulin and A. Fournier. Lights from highlights and shadows. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25 (2), pages 31–38, March 1992.

[21] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In Eugene Fiume, editor, *SIGGRAPH 2001*, pages 117–128, 2001.

[22] Y. Sato, M. D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In Turner Whitted, editor, *SIGGRAPH 97, Computer Graphics Proceedings*, pages 379–388, August 1997.

[23] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with light. In James T. Kajiya, editor, *Proceedings of SIGGRAPH 93*, pages 143–146, August 1993.

[24] R. Shacked and D. Lischinski. Automatic lighting design using a perceptual quality metric. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3) of *Computer Graphics Forum*, pages 215–226. Blackwell Publishing, 2001.

[25] P.-P. Sloan, W. Martin, A. Gooch, and B. Gooch. The lit sphere: A model for capturing NPR shading from art. In B. Watson and J. W. Buchanan, editors, *Proceedings of Graphics Interface 2001*, pages 143–150, 2001.

[26] A. J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *IEEE Visualization 2003*, pages 355 – 362, 2003.

[27] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann, San Francisco, 2002.

[28] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Fifth International Conference on Computer Vision*, pages 902–907, 1995.

[29] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In Alyn Rockwood, editor, *SIGGARPH 99, Computer Graphics Proceedings*, pages 215–224, August 1999.