To appear in *Proc. ICCV 98*

Plenoptic Image Editing

Steven M. Seitz
Computer Sciences Department
University of Wisconsin–Madison
Madison, WI53706
seitz@cs.wisc.edu

Kiriakos N. Kutulakos
Department of Computer Science
University of Rochester
Rochester, NY14607
kyros@cs.rochester.edu

Abstract

This paper presents a new class of interactive image editing operations designed to maintain consistency between multiple images of a physical 3D scene. The distinguishing feature of these operations is that edits to any one image propagate automatically to all other images as if the (unknown) 3D scene had itself been modified. The modified scene can then be viewed interactively from any other camera viewpoint and under different scene illuminations. The approach is useful first as a power-assist that enables a user to quickly modify many images by editing just a few, and second as a means for constructing and editing image-based scene representations by manipulating a set of photographs. The approach works by extending operations like image painting, scissoring, and morphing so that they alter a scene's generalized plenoptic function *in a physically-consistent way, thereby affecting scene appearance from all viewpoints simultaneously. A key element in realizing these operations is a new volumetric decomposition technique for reconstructing an scene's plenoptic function from an incomplete set of camera viewpoints.*

1 Introduction

Image editing programs like Adobe Photoshop provide ways of modifying an object's appearance in a single image by manipulating the pixels of that image. Ultimately, however, one might like to visualize how edits to an object in one image would affect its appearance from other viewpoints and lighting conditions. For instance, consider choosing wallpaper for a room in your house by painting the wallpaper pattern into one of several digitized photographs of the room. As you paint a wall in one image, the pattern appears instantly at the appropriate place in the other images, providing feedback on how the modified room would look from several different viewpoints. Similarly, scissoring out an object (e.g., a vase) from one or two frames of a video walkthrough of a room could remove that object from the entire video by automatically propagating the scissoring operation to the other images. Additional controls could modify scene illumination, reflecting different times of day and varying light source positions, and could modify viewpoint, allowing the effects of image edits to be visualized from viewpoints other than those of the room's original photographs.

In this paper we present an approach that models a scene's appearance from arbitrary viewpoints and illuminations, and allows this appearance to be manipulated via picture editing tools like Photoshop. The key feature of our approach is that it provides a mechanism for (1) allowing pixel changes to one image of a scene to be automatically propagated to all other images in a way that guarantees consistency with a valid 3D shape, and (2) synthesizing arbitrary new views of the edited scene under user-specified lighting conditions. To be realized, any such mechanism requires solving three problems:

- **View synthesis:** how can we create images of the scene from new camera viewpoints?
- **Illumination synthesis:** how can we modify images of the scene to effect changes in scene lighting?
- **Editing:** how can pixel changes due to operations like painting, scissoring, and morphing be propagated across different views of the scene?

A fundamental characteristic of these problems is that they require operating on the space of all views of the scene, rather than just one image. It is therefore convenient to cast them in terms of the *plenoptic function* [1, 2], which encodes scene appearance from all possible viewpoints. Within this framework, we generalize the definition of the plenoptic function to also encode illumination parameters and formulate our goal as one of (1) recovering the scene's plenoptic function from a set of images, and (2) determining how to recalculate the plenoptic function in response to basic image editing operations like painting, scissoring, and morphing. We use the term *plenoptic* to describe image editing operations that modify the plenoptic function and can therefore be propagated to new viewpoints and illuminations.

A key question is how should the plenoptic function be represented in order to enable both synthesis and editing operations. Previous approaches for reconstructing the plenoptic function enabled synthesis of views [2–4] or illuminations [5, 6] but not both. Furthermore, no techniques are currently available for modifying this function in response to image editing operations, unless an *a priori* 3D model is available [7]. For instance, a number of researchers [3, 4, 8] have proposed ray-based representations of the plenoptic function. While these models might in principle be extended to include illumination parameters, the lack of correspondence information does not facilitate plenoptic image editing operations.

In this paper we introduce a new representation of the plenoptic function that is designed to enable both synthesis and editing operations from a set of basis images. This representation, called *plenoptic decomposition*, seeks to exploit the correlated structure of the plenoptic function, by decomposing it into separate shape and radiance components. Plenoptic image editing can then be formulated as a set of operations that act either on the shape or on the radiance component of the plenoptic function. To reconstruct the plenoptic decomposition, we propose a procedure in which 3D space is discretized into a volume of voxels with associated radiance functions that is iteratively carved away to achieve consistency with a set of input images. Unlike previous approaches to shape reconstruction, the carving approach enables changing both viewpoint *and* illumination in the input views.

2 The User View: Plenoptic Editing by Example

Plenoptic image editing is an approach for allowing a user to virtually modify a real scene's appearance in an image-based way by editing any of several photographs of the scene at different positions and orientations. Scene modifications in a plenoptic image editing system occur at two levels—a user level and a system level (Figure 1). From the point of view of the user, all interaction occurs via manipulations to *individual images* using conventional pixel-editing tools. The user simply specifies how one or more images should look by painting and moving pixels until the desired look is achieved. In contrast, system level operations modify a scene's plenoptic function, which affects *all images simultaneously*. Pixel modifications by the user are interpreted as new constraints on scene appearance that induce changes to the plenoptic function and therefore affect every image. In this way, user edits of a single image can be propagated to other images of a scene.

To the user, a plenoptic image editing system appears very similar to current image editing programs like Photoshop. Pixels of one or more images are edited by direct manipulation using a standard suite of painting, scissoring (cut and paste), and warping tools found in many image editing programs. In fact, if only one image is on screen there is no visible difference between a conventional image editing program and the plenoptic version. The difference becomes apparent, however, when two or more images are viewed side by side. Any change to a region or scene in one image is instantly propagated to the corresponding part in the other image(s). For instance, removing a freckle in one of several photographs of a face causes the freckle to disappear simultaneously from all other images. In this way, the propagation mechanism can be used as a kind of *power-assist*—the user can affect many different images of a scene by editing only one or two.

The freckle example illustrates the basic model for plenoptic image editing: a user specifies how regions in one or more images should look *by example*, and the system determines how to consistently propagate the modifications to the other images. This editing-by-example model provides a very powerful way for the user to control scene appearance *plenoptically*, i.e., in all views at once, by editing a small number of images in a direct, intuitive way.

Below we discuss plenoptic versions of some standard image-editing operations. The list is not meant to be comprehensive, but provides examples of what different types of image editing operations can do within a plenoptic framework. We also describe the view and illumination synthesis capabilities provided by our framework. The implementation of these operations is discussed in Section 4.

2.1 Plenoptic Painting

A basic type of image editing operation is to change pixel colors by drawing over an image region with a digital paintbrush. In the plenoptic framework, a paint operation is interpreted as a modification to the material properties of the surface points whose projection coincides with the painted region. The change therefore affects every image of the scene, and properly accounts for differences in visibility between views. The multi-image updates appear in real time, allowing the user to fluidly paint in several images simultaneously by moving a brush over one image. Figure 2 (b) and (f) show images from a real-time plenoptic paint operation in action.

2.2 Plenoptic Scissoring

An image scissoring operation eliminates or extracts a set of regions from an image, often for inclusion in a different image. In contrast, plenoptic image scissoring carves out part of the plenoptic function, causing a corresponding region to be extracted in every image. Scissoring out the image region therefore has the effect of cutting out the portion of the scene that projects to that image region.

Plenoptic scissoring enables some interesting effects that are not possible with regular scissoring. For instance, it is possible to "see through" objects in an image by scissoring them out and exposing what lies behind. This capability is shown in Fig 2 (g) and is achieved by extrapolating the appearance of hidden surfaces from other images in which those surfaces are visible, using the derived plenoptic model. The extrapolation occurs automatically whenever the user performs a scissoring operation.

2.3 Plenoptic Morphing

Image warping or *morphing* [9, 10] is a popular way of producing shape changes and animations from one or more images. Although multi-image morphs can be performed in the plenoptic framework, we restrict our attention to the case in which a single image is warped by displacing individual pixels using a 2D motion flow field. Instead of warping pixels, a plenoptic morph warps the underlying 3D scene so as to be projectively consistent with the warped image, as shown in Fig. 2 (h). These effects can be thought of in terms of warping *rays* in space. Consider, for example, the ray that originates at a camera's center and passes through the image plane at a particular pixel. Moving this pixel corresponds to moving all points along the ray to coincide with the ray passing through the destination pixel.

2.4 New View Generation

In addition to propagating changes between images, the plenoptic framework can generate arbitrary new views of a scene under different illuminants by evaluating the recovered plenoptic function at new user-specified viewpoints and light-source configurations. The synthetic views automatically incorporate the modifications incurred by user edits to the images, since these edits induce changes to the plenoptic function.

The ability to generate new views is also useful for edit operations, because it allows the user to interactively choose a good image for editing. For instance, a flat surface can be rotated to a front-on view [11, 12] to facilitate painting and avoid foreshortening effects. Similarly, scissoring a region is easier when the entire region is visible in a single image.

3 Behind the Scenes: Plenoptic Decomposition

In order to generate new views of an object and to perform plenoptic editing operations, we must first model the plenoptic function in a way that makes it easy to (1) generate new samples of the plenoptic function (i.e., images), and (2) modify the function via changes in a single image. For this purpose, we use a novel plenoptic reconstruction method called *plenoptic decomposition*, which decomposes the plenoptic function into shape and radiance components (Figure 3(a)). An advantage of this plenoptic function representation is that any 2D plenoptic image editing operation can be immediately transformed into an operation in 3D. Furthermore, because local changes to an image require only local changes to the representation, plenoptic editing operations can be propagated very efficiently between images.

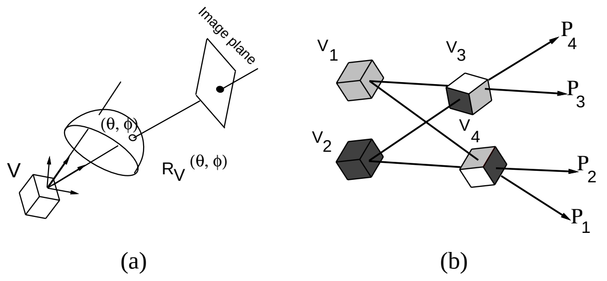

Figure 3. Plenoptic decomposition. (a) Every ray in space can be thought of as emanating from a unique voxel V. The shape component in the plenoptic decomposition is the set of all such voxels. The radiance at V is a function $R_V(\theta, \rho)$ that describes how light emanating from the point flows in a given direction. (b) Suppose that all rays emanating from voxels V_1, V_2 are gray and black, respectively. Then, there are two distinct ways of "explaining" the rays $P_1, \ldots, P_4$ in terms of shape and radiance: (1) P_1, P_3 and P_2, P_4 originate from the "gray" and "black" voxels V_1 and V_2, respectively, or (2) P_1, P_2 and P_3, P_4 originate from V_4 and V_3, respectively. In the latter interpretation, V_3 and V_4 have non-constant radiance, i.e., their color depends on the viewing position. If a constant radiance model were enforced, V_3 and V_4 would be deemed inconsistent with the radiance model and carved away during plenoptic decomposition.

The plenoptic function of a 3D scene describes the flow of light along every oriented ray in space, and encodes the scene's appearance from every direction [1–4]. While the plenoptic function is determined uniquely by the 3D surfaces in a scene and their reflectance properties, we can generate the same plenoptic function by combining many different shapes and radiance functions[1] (Figure 3(b)). Plenoptic decomposition resolves this ambiguity by enforcing consistency with an *a priori*-specified scene radiance model. This consistency enforcement leads to a representation for the plenoptic function that is particularly suitable for plenoptic reconstruction and editing.

To arrive at this representation, plenoptic decomposition proceeds in three steps. The first step involves defining a cube of voxels $\mathcal{V} = \{V_1, \ldots, V_k\}$ that encloses the scene. The image projections of every voxel V_i on the surface of this cube define correspondences between pixels in the input images. Furthermore, the color and intensity of corresponding pixels can be thought of as samples of the radiance function of a hypothetical scene point positioned at V_i. The second step of the method recovers the shape component of the plenoptic function representation

[1] Holographic imaging [13] is one notable application where this ambiguity is put into practical use: it relies on our *in*ability to distinguish views of flat holographic images from views of objects that are truly 3D.

by carving away from $\mathcal{V}$ all voxels whose projections are not consistent with the *a priori*-specified radiance model. Upon completion of this step, the reconstructed shape component is a volume of uncarved voxels that conform to the chosen radiance model. In the final step of the method, the radiance function of every uncarved voxel in $\mathcal{V}$ is recovered from the voxel's projections in the input images.

3.1 Voxel Carving

As outlined above, our strategy for plenoptic decomposition computes an estimate of the object's shape by incrementally carving away from a block of voxels, using the coherence of emitted light as a criterion for voxel elimination. The main idea is to define a local radiance model (e.g., ambient, Lambertian) and to carve away voxels that do not conform to this model based on the pixel correlation of their image projections. In order to compute these projections, we assume that the input viewpoints are known and that the visible scene lies entirely outside of the *camera volume*, i.e., the convex hull of the camera centers. The carving algorithm takes advantage of a voxel enumeration strategy that visits voxels in "depth-order" to account for occlusions [14]. Here we employ this enumeration strategy to facilitate plenoptic decomposition, i.e., recovery of shape *and* parametric radiance functions.

The voxel carving algorithm operates as follows: the scene is initialized to a solid block of voxels. This block should be large enough to fully enclose the area spanned by the object or scene to be reconstructed. The voxels are then processed, one at a time, by determining how well their image projections conform to a fixed model of scene radiance. Voxels whose correlation falls below a threshold are carved away (eliminated) from the volume. The voxels that remain at the end represent the shape component of the plenoptic decomposition. The steps are as follows:

1. Enumerate the voxels $\{V_1, \ldots, V_k\}$ in order of increasing distance from the camera volume, as in [14]
2. For each voxel V_i, $i = 1, 2, \ldots, k$:
   (a) project V_i to the input images; let $C_1, \ldots, C_n$ be the colors of the *unmarked* image pixels to which V_i projects
   (b) evaluate the coherence of colors $C_1, \ldots, C_n$ using Eq. (2) (see Section 3.2); if the coherence metric is less than some threshold c, mark these pixels and output V_i.

3.2 Radiance Modeling

A key component of plenoptic decomposition is establishing a model for voxel radiance, which is used both in the carving and radiance reconstruction steps. Ideally, the radiance model should be chosen to match that of the observed scene. In practice, a highly-accurate model for the light radiating from a physical 3D scene is rarely available, and image quantization, sampling, and noise will inevitably lead to images that do not conform exactly to a single radiance model. To account for these effects, we define the radiance R of individual voxels to be the sum of two components,

$$R = R_{\text{ideal}} + R_{\text{res}}, \tag{1}$$

where R_{ideal} is a parameterized model for the voxel's ideal radiance and R_{res} captures spatially-localized deviations from this model that occur in one or more of the input images. Both radiance components are defined to be functions over the sphere of relative orientations between the voxel and a camera-centered reference frame (Figure 3(a)); this ensures that the plenoptic function is represented in terms of *observable* quantities (camera position) rather than the underlying physical parameters giving rise to it (e.g., surface normals, BRDF, positions of light sources).

Once the model for a voxel's radiance is recovered, voxel consistency is established by an equation of the form

$$\|\vec{C} - \vec{C_m}\| < c \tag{2}$$

where $\vec{C}$ is a vector that holds the colors at the voxel's projection in multiple images, $\vec{C_m}$ holds those predicted by the ideal radiance model, and c is a threshold term.

3.2.1 Lambertian Radiance

To account for shading effects due to changes in the relative positions of an object and light sources, we use a Lambertian model for modeling a voxel's ideal radiance [5, 15, 16]:

$$C(M) = \left[I_{\text{amb}} + \sum_i \alpha_i \, \vec{n}_i M \vec{l}_i \right] C_{base} \tag{3}$$

The model treats each voxel as an independent Lambertian surface that is illuminated by multiple light sources at infinity, has color C_{base}, and has normal $\vec{n}$. The advantages of this model are that (1) radiance can be recovered and expressed directly as a function of the rotation matrix M that describes the scene's orientation relative to the camera, (2) it can be reconstructed independently for each voxel, (3) it can better account for illumination variations at different parts of an object, and (4) it can be adapted to enforce local illumination coherence constraints. In practice, neither the positions of the light sources nor the surface normal of each voxel are known. We overcome this problem with the help of a linear method that expresses radiance as a function of the object's (known) rotation:

$$C(M) = \left[I_{\text{amb}} + \sum_{i=1,2,3} \sum_{j=1,2,3} m_{ij} X_{ij} \right] \overline{C} \tag{4}$$

where (m_{ij}) are the elements of the rotation matrix, $\overline{C}$ is the mean value of $C(M)$, and X_{ij} are constants that are different for each voxel and are computed by the method. Intuitively, Eq. (4) expresses the radiance of a voxel directly in terms of image measurements, without attempting to recover the normal and light source vectors as in traditional photometric stereo techniques [15]. Recovering a model for the radiance of a voxel then involves solving a linear system of equations for the unknowns I_{amb} and X_{ij} in terms of known m_{ij} and observed $C(M)$.[2]

3.2.2 Modeling Residuals

In plenoptic decomposition, radiance residuals are used to ensure that local radiance variations are approximated accurately for views close to the input images [17]. Residual modeling is an instance of scattered data approximation on the sphere [18]—a rich literature on the topic exists, partly motivated by the problem of BRDF estimation [19, 20]. Rather than treating the problem in its full generality, we consider a simpler approach that can be used to model residuals for views taken along a single-axis rotation of the object. This allows us to further reduce the dimensionality of the approximation problem and simplify computations. The process consists of (1) constructing a one-dimensional multi-resolution representation for the residuals $R_{\text{res}}(\theta)$ by means of a Haar wavelet basis, and (2) propagating this function to all points on the orientation sphere according to $R_{\text{res}}(\theta, \phi) = R_{\text{res}}(\theta)\cos(\phi)$. This heuristic propagation step attempts to preserve the structure of the residual function throughout the sphere, while reducing the contribution of radiance residuals for viewpoints away from the original images.

4 Implementation

This section briefly describes the implementation of the plenoptic image editing operations shown in Figure 2. Each operation changes either the shape or radiance component of the plenoptic decomposition, but not both. This property simplifies the implementation and enables operations to be performed more efficiently. The last part of this section describes our experimental setup and the acquisition of images.

4.1 Painting

Painting is the simplest of the plenoptic image editing functions because it changes only the radiance function of scene voxels without any modification to shape. Propagating a painted pixel requires first determining the voxels of the plenoptic decomposition that correspond to that pixel and modifying their radiance functions. In our current implementation, the voxels are simply assigned an isotropic radiance corresponding to the painted pixel color. The change is then propagated by projecting the voxel into each image in which it is visible and re-coloring corresponding pixels in those images.

Plenoptic painting can be performed in real-time by precomputing the mapping between pixels in each image and voxels in the plenoptic decomposition. Our implementation allows a user to paint in several images simultaneously, using this technique.

4.2 Scissoring

Image scissoring cuts out a set of pixels from an image. Similarly, plenoptic scissoring removes a set of voxels from the plenoptic decomposition. One option is to remove the set of voxels that project unoccluded to the affected pixels. This may expose new voxels in the scissored image, behind those that were removed. Alternatively, scissoring can remove all voxels that project to the affected pixels, whether or not they are visible. The latter method was used to generate the images in Figure 2 (c) and (g).

Performing the propagation requires masking out pixels in each image that correspond to the projection of voxels removed by the scissoring operation. These pixels are then filled in by rendering the scissored plenoptic model and copying these pixels from the rendered image.

4.3 Morphing

As described in Section 2, an image morph induces a warping of scene rays. Consider the set of rays passing from a camera center through the image plane. An image morph deforms the image plane, causing these rays to move with it. In turn, the motion of a ray moves all scene voxels that lie on the ray. While the motion of rays is determined, the motion of voxels along rays is not. Our implementation of plenoptic image morphing fixed this variable by constraining voxels to move parallel to the image plane and used Beier and Neely's method [9] to generate image warps. Morph propagation is achieved by using the projected voxel displacement to define image warps in new views. Voxels that become unoccluded as a result of the morph are rendered directly, as described in Section 5.

4.4 Image Acquisition

Calibrated images were acquired by rotating an object on a software-controlled pan-tilt head in front of a stationary camera. The camera was raised slightly above the object to be compatible with the ordinal visibility constraint [14]. The illumination was fixed relative to the camera, causing changes in shading as the object rotated. This effective variation in illumination enabled computation of the Lambertian coefficients as described in Section 3.2.

[2] A consequence of Eq. (4) is that a minimum of 10 views are needed to obtain a radiance model for each voxel independently. In principle, the minimum number of views can be further reduced by recovering the radiance of multiple voxels simultaneously and by making explicit the nonlinear relations between the equation's seven independent parameters.

5 Rendering

Once the plenoptic decomposition has been computed for an object, that object can be rendered from different camera positions by (1) evaluating the radiance function for each voxel V corresponding to the desired viewpoint and (2) projecting that voxel into the image. Alternatively, the illumination can be artificially changed by projecting the voxels into a viewpoint that is different than the one at which the radiance functions are evaluated.

Figure 4 compares renderings of the plenoptic decomposition recovered from 21 images of a 360 degree rotation of a dinosaur toy with different radiance models. Adding the Lambertian components yields a significant overall improvement, and the wavelet coefficients restore fine detail, as seen in the blowup. Figure 5 shows the extrapolation capabilities of the system using the recovered plenoptic decomposition from 21 views of a real flower. These images demonstrate the ability of the algorithm to synthesize both changes in camera position and illumination.

6 Conclusions

Plenoptic editing puts forth a new class of image editing operations that allow edits in one image to be automatically propagated to all possible views of an object, in a physically-consistent way. We showed that these operations can be realized with the help of three novel methods for recovering, editing, and rendering an object's plenoptic function. At the heart of our approach is *plenoptic decomposition*—a new framework for decomposing an object's plenoptic function into separate shape and radiance components. This framework enables reconstruction from a sparse set of input images, taken under varying viewpoints and illuminations, and allows plenoptic image edits to occur in a direct and efficient manner.

Our radiance modeling method is currently restricted to Lambertian surfaces with light sources at infinity and does not account for shadows. The method could be generalized, however, to cope with other local reflectance models. Unfortunately, accounting for shadows is more difficult due to the global interactions between surfaces and light sources.

The propagation mechanism relies on voxel carving to obtain accurate pixel correspondence information. Incorrect correspondences can cause distracting errors in propagation, e.g., applying paint to one image changes the wrong part of a different image. Like most reconstruction techniques, voxel carving is susceptible to shape errors in low-contrast, untextured regions of the scene. We are currently investigating the possibility of adding an interactive correction mechanism in which a user can modify a propagation by manually painting in the desired changes to a second or third image.

References

[1] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computation Models of Visual Processing* (M. Landy and J. A. Movshon, eds.), MIT Press, 1991.

[2] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," in *Proc. SIGGRAPH'95*, pp. 39–46, 1995.

[3] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. SIGGRAPH '96*, pp. 31–42, 1996.

[4] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proc. SIGGRAPH'96*, pp. 43–54, 1996.

[5] A. Shashua, *Geometry and Photometry in 3D Visual Recognition*. PhD thesis, MIT, 1992.

[6] P. N. Belhumeur and D. J. Kriegman, "What is the set of images of an object under all possible lighting conditions," in *Proc. CVPR*, pp. 270–277, 1996.

[7] P. Hanrahan and P. Haeberli, "Direct WYSIWYG painting and texturing on 3D shapes," in *Proc. SIGGRAPH 90*, pp. 215–223, 1990.

[8] K. N. Kutulakos, "Shape from the light field boundary," in *Proc. CVPR*, pp. 53–59, 1997.

[9] T. Beier and S. Neely, "Feature-based image metamorphosis," in *Proc. SIGGRAPH 92*, pp. 35–42, 1992.

[10] S. M. Seitz and C. R. Dyer, "View morphing," in *Proc. SIGGRAPH 96*, pp. 21–30, 1996.

[11] D. Wilkes and J. K. Tsotsos, "Active object recognition," in *Proc. CVPR*, pp. 136–141, 1992.

[12] K. N. Kutulakos and C. R. Dyer, "Recovering shape by purposive viewpoint adjustment," *Int. J. Computer Vision*, vol. 12, no. 2, pp. 113–136, 1994.

[13] S. Benton, "Survey of holographic stereograms," in *Processing and Display of 3D Data*, Proc. SPIE, 1983.

[14] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *Proc. CVPR*, pp. 1067–1073, 1997.

[15] R. J. Woodham, Y. Iwahori, and R. A. Barman, "Photometric stereo: Lambertian reflectance and light sources with unknown direction and strength," Tech. Rep. 91-18, Univ. of B.C., Lab. for Computational Intelligence, August 1991.

[16] R. Epstein, A. L. Yuille, and P. N. Belhumeur, "Learning object representations from lighting variations," in *Object Rep. in Computer Vision II* (J. Ponce, A. Zisserman, and M. Hebert, eds.), pp. 179–199, Springer-Verlag, 1996.

[17] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," in *Proc. SIGGRAPH'96*, pp. 11–20, 1996.

[18] G. Nielson, "Scattered data modeling," *IEEE Computer Graphics and Applications*, pp. 60–70, 1993.

[19] P. Schroder and W. Sweldens, "Spherical wavelets: Efficiently representing functions on the sphere," in *Proc. SIGGRAPH'95*, pp. 161–172, 1995.

[20] F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg, "A global illumination solution for general reflectance distribution functions," in *Proc. SIGGRAPH'91*, pp. 187–196, 1991.

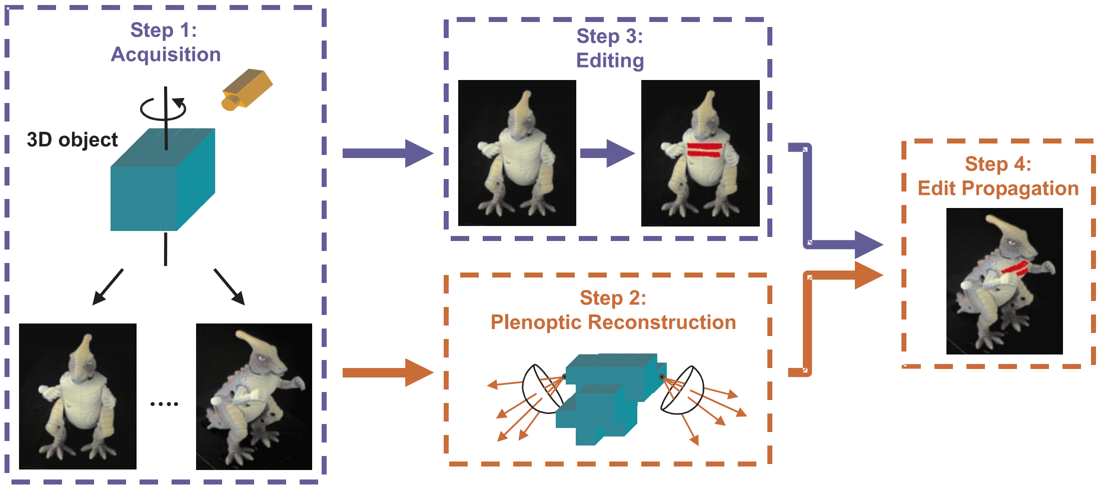

Figure 1. Overview of our system for plenoptic reconstruction and editing. From the point of view of the user, system operation involves two steps (blue boxes and transitions): an image acquisition step (Step 1), in which multiple images of a scene are acquired for different viewpoints and illumination conditions, and a scene editing step (Step 3) that allows a scene's appearance to be manipulated by editing individual images of the scene. At the system level (red boxes and transitions), the acquired images are used to recover a representation for the scene's plenoptic function (Step 2). This representation consists of a shape component (a set of voxels in space) and a radiance component (the color and intensity of rays reflected from every voxel in every direction). Once this representation is recovered, user-specified edits to a single image are propagated automatically to all views of the scene (Step 4).

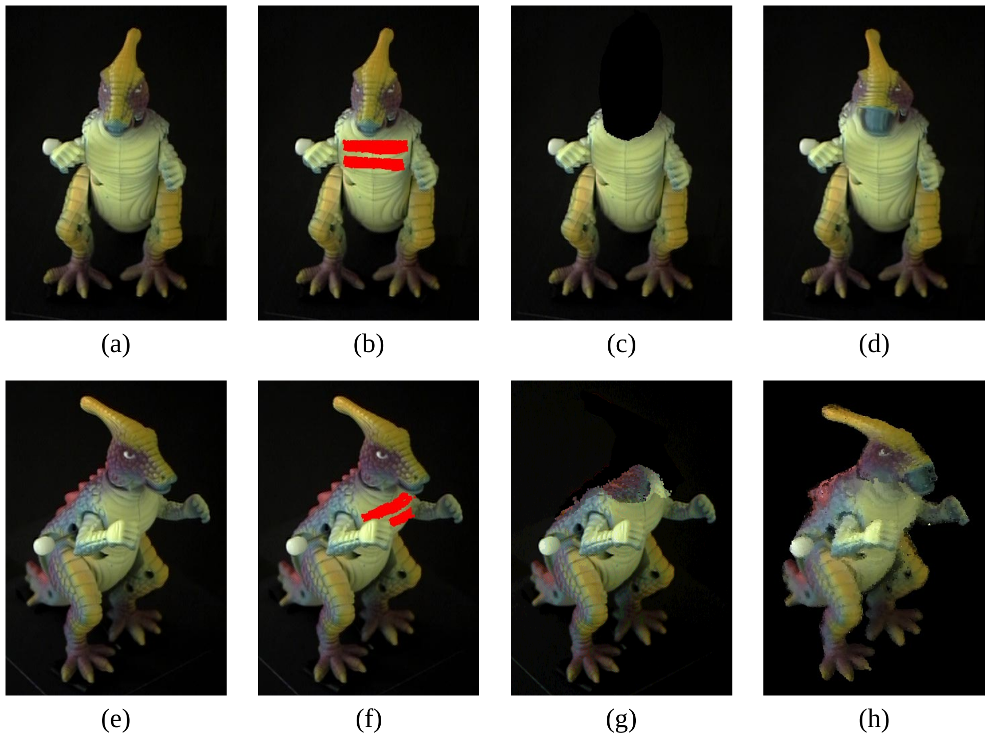

Figure 2. Examples of plenoptic image editing operations applied to photographs of a dinosaur toy. (b) - (d) show image painting, scissoring, and morphing operations, respectively, applied to image (a). (f) - (h) show images that were automatically generated by propagating these respective editing operations to image (e). Observe that the propagation properly accounts for differences in visibility between the two views—part of the painted area is correctly occluded by the dinosaur's right hand in image (f), and cutting off the head in image (c) exposes surfaces in image (g) that were not visible in the original image (e). These new surfaces are *synthesized* from other viewpoints so that (g) represents a composite of a real photograph with synthesized image regions.

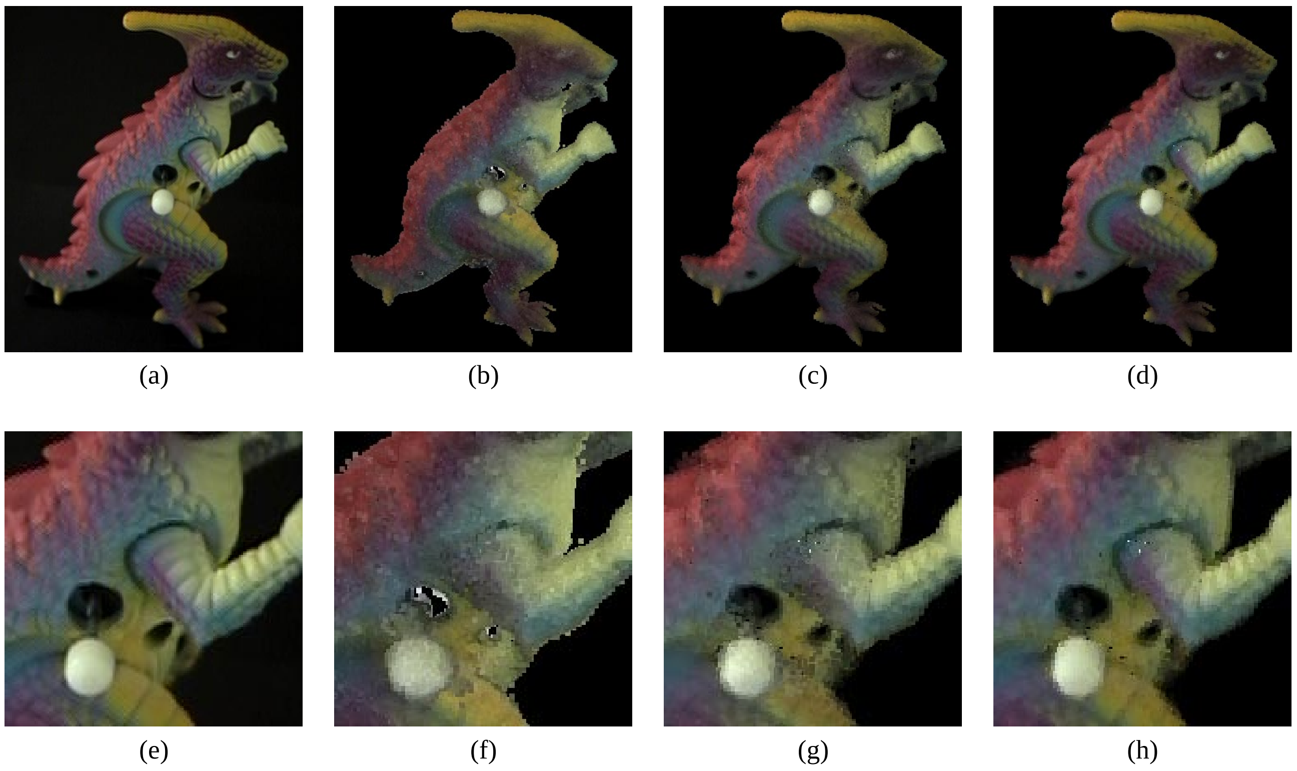

Figure 4. Plenoptic reconstruction. (a) shows one of 21 original images of a toy dinosaur. (b)-(d) show reconstructions with different radiance models: (b) was generated using an ambient (isotropic model); (c) used the Lambertian model, and (d) included a residual model with a maximum of 20 wavelet coefficients for each voxel. (e)-(h) show detail in a subregion of images (a)-(d) respectively.

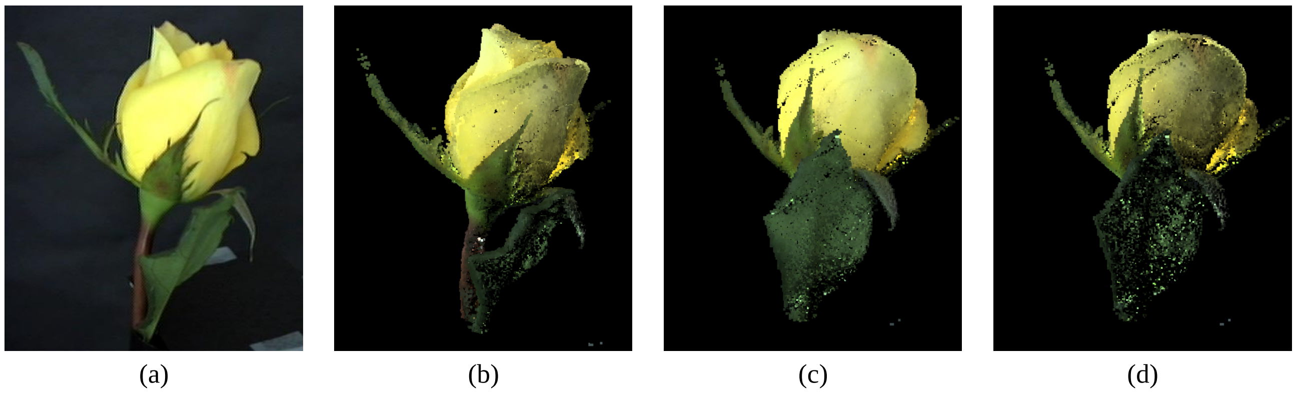

Figure 5. View and illumination synthesis. (a) shows one of 21 original images of a rose. (b)-(f) illustrate the extrapolation capabilities of our system—the views were synthetically generated and did not have counterparts in the input sequence: (b) shows the rose from the same viewpoint but with a different illumination; (c) and (d) show a new view of the rose from below, illuminated as in (a) and (b) respectively. The lighting effects in (b) and (d) were produced by evaluating the voxels' radiance functions for viewing parameters different than those used to project the voxels.