Plenoptic Image Editing

Steven M. Seitz
Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195-2350 USA
seitz@cs.washington.edu

Kiriakos N. Kutulakos
Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4 CANADA
kyros@cs.toronto.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 *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.

Keywords: image-based scene representations, multiple-view geometry, plenoptic function, image manipulation, 2D painting and drawing, view synthesis, image-based rendering

This work was performed while Steven Seitz was with the Robotics Institute, Carnegie-Mellon University and Kiriakos Kutulakos was with the Departments of Computer Science and Dermatology, University of Rochester.

1 Introduction

Image editing programs like Adobe Photoshop [1] 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* [2, 3], 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.

Using the plenoptic function framework as a starting point, our approach can be thought of as operating on three conceptual levels. On the *representational level* we introduce plenoptic decomposition as a novel way to represent the plenoptic function and the way it varies under different illumination conditions. On the *computational level* we rely on a voxel-based algorithm that computes a scene's plenoptic decomposition from the input images. On the *interface level* we use a Photoshop-like interface to perform plenoptic edits while hiding all image-derived scene representations from the user.

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 [3–11] or illuminations [12, 13] but not both. Furthermore, no techniques are currently available for modifying this function in response to image editing operations. For instance, a number of researchers [7, 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 operations that involve image editing. Performing image editing operations within a ray-based representation is difficult for two reasons. First, local image edits can affect object appearance from disparate views and may therefore require global changes to a ray-based representation. Second, the lack of correspondence information makes it difficult to propagate editing operations between images.

To overcome these limitations, we propose a new representation of the plenoptic function that is designed to enable both synthesis and editing operations from a set of input images. This representation, called *plenoptic decomposition*, seeks to exploit the correlated structure of the plenoptic function [14, 15], 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 compute the plenoptic decomposition from a set of input images, we propose a procedure in which 3D space is first discretized into a volume of voxels with associated radiance functions and then iteratively carved away to achieve consistency with a set of input images. Unlike previous approaches to shape reconstruction, this voxel-based approach enables changing both viewpoint *and* illumination in the input views. In addition, this decomposition approach has significant advantages over previous correspondence-based approaches to image-based rendering. Previous approaches have relied on stereo vision techniques [3, 5, 9, 11, 16–18] or silhouette-based techniques [19–22] to derive pixel correspondence information. Both types of methods have serious weaknesses—existing stereo techniques require that the input cameras be close together. On the other hand, silhouette and other contour-based methods fail at concavities. In contrast, plenoptic decomposition enables accurate, dense pixel correspondence information to be obtained from cameras that are widely distributed around a scene.

We emphasize that the goal of this work is to manipulate scene appearance in a 2D image-based manner, by editing pixels rather than surfaces. In this respect, our approach departs from 3D editing systems [23, 24] which seek to provide a three-dimensional interface for editing operations. Another key difference is that plenoptic image editing does not require a priori knowledge of scene shape and can therefore operate on photographs of real scenes.

While our approach involves computing a 3D voxel-based model of the scene, this model is used only as a means for propagating pixel edits and illumination changes between different views of the scene. Indeed, the existence of a 3D model is invisible to the user, who operates entirely on 2D images. All user edits and propagated edits appear as image modifications, in which a subset of pixels in different images are changed in response to painting, scissoring, or morphing operations. Two advantages of this *image-based* approach are (1) the quality of the original images is preserved, (2) accurate 3D geometry is generally needed only in the area of the editing operation, and (3) we have found the Photoshop-style interface simple to use and easier to learn than a 3D CAD-style interface.

The rest of the paper is organized as follows. The next section describes how a plenoptic image editing system appears to the user. Section 3 motivates the use of a shape-radiance model for representing a scene's plenoptic function and outlines a method for its reconstruction from a set of input images. Section 4 then describes how the recovered representation can be used to synthesize images of the scene under new viewpoints and illuminations and to propagate image edits across these different views. Section 5 presents results from our experimental plenoptic image editing system as applied to images of real 3D scenes.

2 The User's View: 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 manner 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 image pixels 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 a subset of the pixels in every image. In this way, user edits of a single image can be propagated to other images of a scene. Importantly, the original images are modified only in regions that are affected by the propagation of the editing operation.

To the user, a plenoptic image editing system appears very similar to current image editing programs

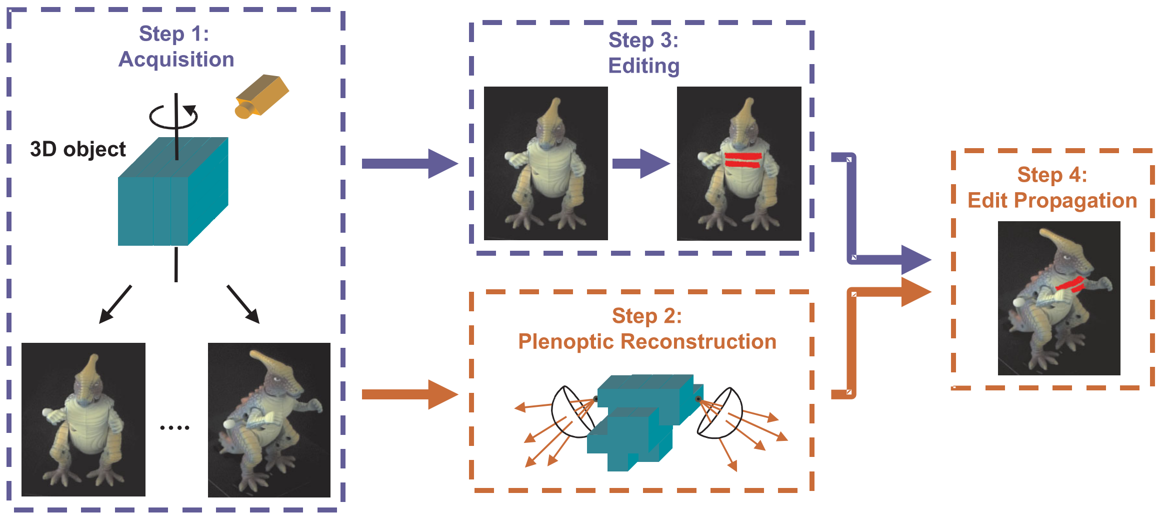


Figure 1: Overview of our system for plenoptic reconstruction and editing. From the point of view of the user, system operation involves two steps (blue boxes and transitions): an image acquisition step (Step 1), in which multiple images of a scene are acquired for different viewpoints and illumination 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).

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 an 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 all 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 [1]. In the plenoptic framework, a paint operation is interpreted as a modification to the material properties of the surface points whose projections coincide 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 (e) 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 [25, 26]. 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.

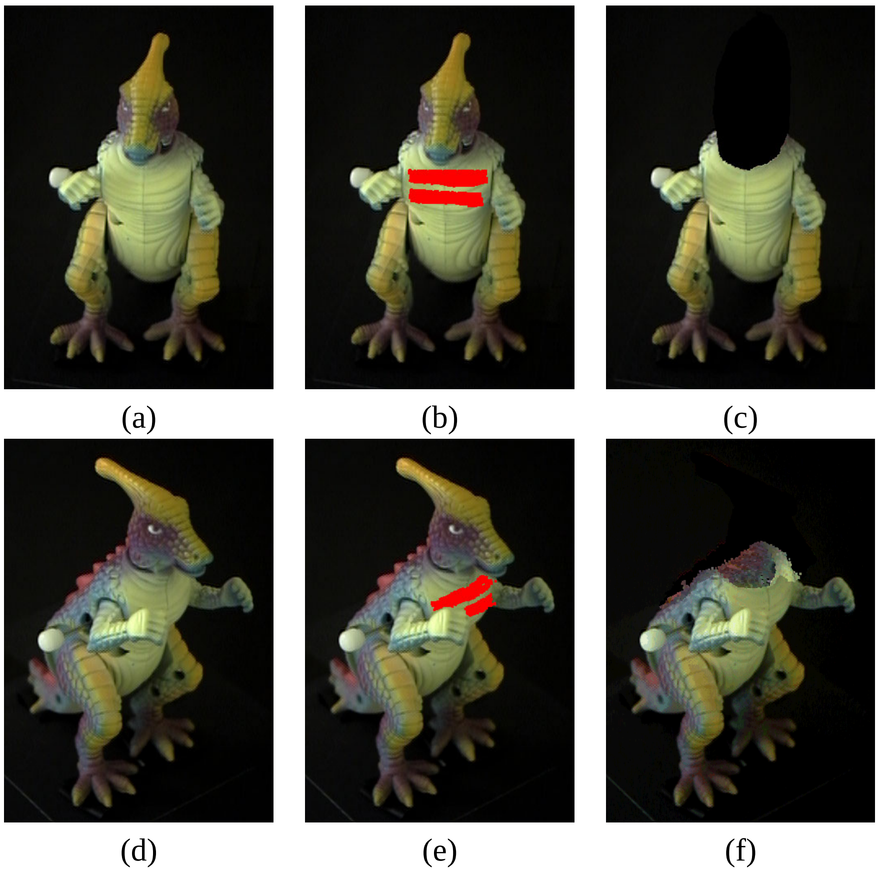


Figure 2: Examples of plenoptic image editing operations applied to photographs of a dinosaur toy. (b) and (c) show image painting and scissoring operations, respectively, applied to image (a). (e) and (f) show images that were automatically generated by propagating these respective editing operations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

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 (f) 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* [27] is a popular technique for 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 an interactively-specified 2D motion flow field. Instead of only warping pixels in that single image, however, a plenoptic warp induces a warp in all input images to make them projectively consistent with a well-defined "warped" 3D scene. This effect 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. This ray motion defines a warp in every input image corresponding to the projected motion of all points along the ray.

2.4 View Synthesis

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 [28, 29] 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) under varying illumination conditions, 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.

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 [2, 7, 8]. In its original formulation [2], it was a function of seven parameters that captured the variation of scene appearance with respect to camera position, direction of the optical axis, wavelength and time. To model appearance variations that result from changes in the relative position of a scene and its light sources, we use an alternate parameterization in which the time parameter is replaced by a 3×3 rotation matrix M. This matrix controls the scene's relative orientation with respect to a collection of stationary light sources at infinity.

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

[1]Holographic imaging [30] 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.

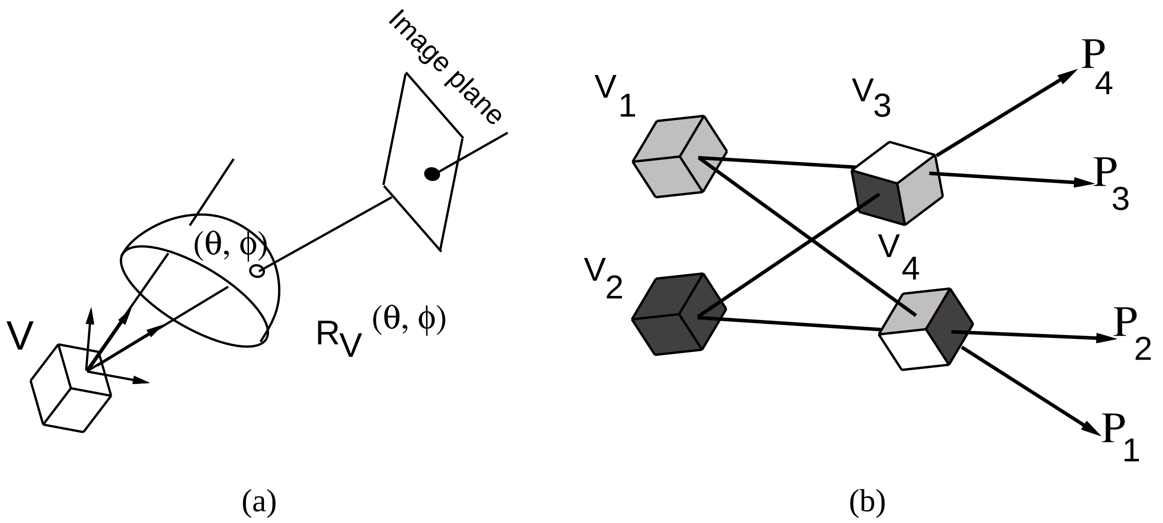


Figure 3: Plenoptic decomposition. (a) Every ray in space can be thought of as emanating from a unique voxel V. The shape component in the plenoptic decomposition is the set of all such voxels. The radiance at V is a function $R_V(\theta, \phi)$ that describes how light emanating from the point flows 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.

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 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 third and final step of the method, the radiance function of every uncarved voxel in $\mathcal{V}$ is recovered from the voxel's projection in the input images.

3.1 Voxel-Based Reconstruction

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 algorithm takes advantage of a voxel enumeration strategy that visits voxels in "depth-order" to account for occlusions [31]. Here we employ this enumeration strategy to facilitate plenoptic decomposition, i.e., recovery of shape *and* parametric radiance functions. We also note that plenoptic decomposition may also be used in conjunction with the *Space Carving* technique [32] which, unlike [31], is applicable for completely general camera configurations.

The voxel-based 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 layer 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 [31].

2. Perform the following two operations 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 and let $\mathrm{M}_1, \ldots, \mathrm{M}_n$ be the matrices describing the scene's orientation, respectively;

(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 radiance modeling. Radiance modeling is used to recover a model for the radiance of each voxel and to decide which voxels to carve. Ideally, the radiance model should be chosen to match that of the observed scene. This model gives a principled way to answer three questions:

- **Coherence evaluation:** given a voxel V and a collection of images it projects to, is V's radiance consistent with the radiance model of scene points?
- **Radiance reconstruction for generating new views:** given a voxel V in the scene's plenoptic decomposition, a new camera position, and a rotation matrix **M** what is the color of V's projection for the new viewpoint and lighting conditions?
- **Edit propagation for plenoptic painting:** how does a painting operation in one or more images affect the color of re-painted voxels in all remaining views?

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 model these effects, we define the radiance R of individual voxels to be the sum of two components,

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

where R_{ideal} is a parameterized model for the voxel's ideal radiance that can be evaluated for every direction when its parameters are known, and R_{res} captures spatially-localized deviations from this

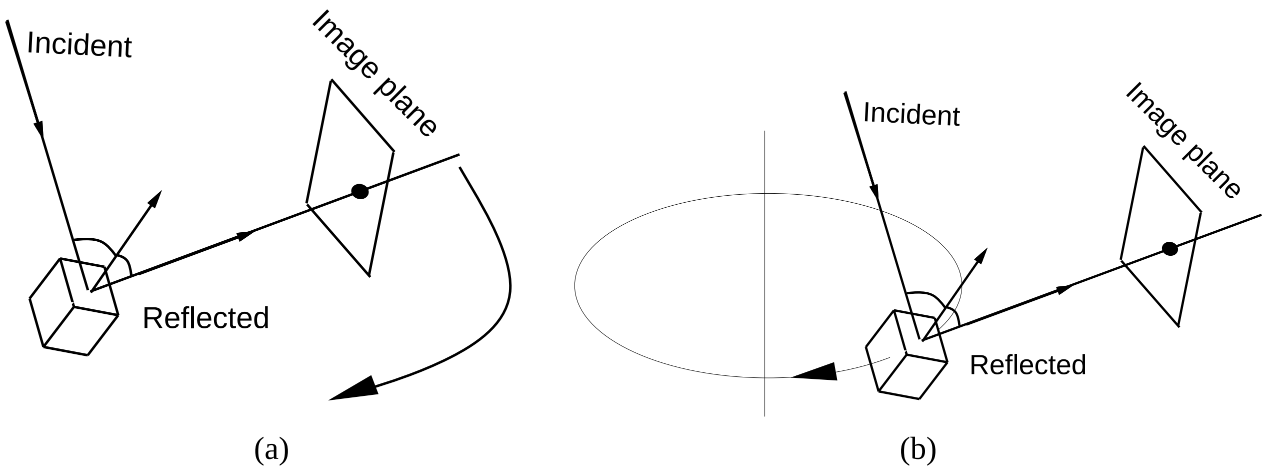


Figure 4: Image acquisition configurations. (a) Images are acquired by moving the camera; the relation between voxels and light sources remains fixed in all images (i.e., M is constant) and surface shading is the same in all images. (b) Images are acquired by rotating the object; the rotation changes the orientation of the voxel relative to the camera and the light sources, and hence changes surface shading. A distinct matrix M is associated with each input image.

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 (radiance) 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[2]

$$\left(\sqrt{\frac{1}{n} \sum_{j=1}^{n} [R(\mathrm{M}_j) - C_j]^2} \right) < c \tag{2}$$

where n is the number of images in which the voxel is visible, $R(\mathrm{M}_j)$ is the color predicted by the voxel's radiance model for the j-th image, C_j is the voxel's observed color in that image, and c is a threshold term.

[2]For simplicity, in this equation we parameterize the radiance R by the rotation matrix M. Every matrix M can be trivially be converted to the (θ, ϕ) representation for any given voxel.

3.2.1 Lambertian Radiance

To account for shading effects due to changes in the relative position of an object and a light source, (Figure 4), we use a Lambertian model for modeling a voxel's ideal radiance [12, 33, 34]:

$$R_{ideal}(\mathrm{M}) = \left[e_{amb} + \rho \sum_{i=1}^{m} e_i \; N \mathrm{M} L_i \right] R_{base} \tag{3}$$

The model treats each voxel as an independent Lambertian surface that is illuminated by multiple light sources at infinity, has color R_{base}, surface normal N, and reflectance ratio ρ. The m light sources have orientations L_i and intensities e_i. 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 position 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 rotation matrix describing the object's (known) rotation. More specifically, by expanding Eq. (3) we obtain:

$$\begin{aligned} R_{ideal}(\mathrm{M}) &= R_{base} \left\{ e_{amb} + \rho \sum_{i=1}^{m} \left(e_i \begin{bmatrix} n^1 & n^2 & n^3 \end{bmatrix} \begin{bmatrix} \mu^{11} & \mu^{12} & m^{13} \\ \mu^{21} & \mu^{22} & \mu^{23} \\ \mu^{31} & \mu^{32} & \mu^{33} \end{bmatrix} \begin{bmatrix} l_i^1 \\ l_i^2 \\ l_i^3 \end{bmatrix} \right) \right\} &(4) \\ &= R_{base} \left[e_{amb} + \rho \sum_{i=1}^{m} e_i \left(\sum_{j=1}^{3} \sum_{k=1}^{3} n^j l^k \mu_{jk} \right) \right] &(5) \\ &= R_{base} \left[e_{amb} + \sum_{j=1}^{3} \sum_{k=1}^{3} \mu_{jk} \left(\rho \sum_{i=1}^{m} e_i n^j l^k \right) \right] . &(6) \end{aligned}$$

We can therefore express the radiance of a voxel using the equation

$$R_{ideal}(\mathrm{M}) = R_{base} \left[e_{amb} + \sum_{j=1}^{3} \sum_{k=1}^{3} \mu_{jk} x_{jk} \right] , \tag{7}$$

where (μ_{jk}) are the elements of the rotation matrix, R_{base} is an (R,G,B) triplet that encodes the voxel's color,[3] and x_{jk} are constants that are different for each voxel. Recovering a model for the radiance of a voxel therefore involves solving a linear system of equations for the unknowns e_{amb} and x_{jk} in terms of known m_{jk} and observed R_{base}. As such, Eq. (7) expresses the radiance of a voxel directly in terms of image measurements, without requiring the recovery of the normal and light source vectors as in traditional photometric stereo techniques [33].

Our use of Eq. (7) for modeling voxel radiance has two important consequences. First, a minimum of ten views is needed to obtain a radiance model for each voxel independently. While this number of views may appear large, it is precisely in such cases that our plenoptic editing technique is especially useful, i.e., when a large collection of input views of a 3D scene must be edited simultaneously. In principle, the minimum number of views can be reduced by recovering a radiance model for multiple voxels simultaneously and by making explicit the non-linear relations between Eq. (4)'s seven voxel-specific independent parameters. Second, the x_{jk}'s recovered for a given voxel implicitly define a set of light source directions that are consistent with the observed colors at the voxel's projection. Unfortunately, when this equation is solved independently for multiple voxels, there is no guarantee that the light source directions implicitly defined by the x_{jk}'s will be the same across all voxels. Hence our method does not guarantee that the assignment of radiance models to voxels is globally consistent—achieving global consistency is a topic of future work.

3.2.2 Modeling Residuals

In plenoptic decomposition, radiance residuals are used to ensure that a voxel's local radiance variations are approximated accurately for views close to the input images (Eq. 1) [9]. These variations become significant when the voxel's appearance (i.e., color and intensity) in one or more input images differs from the one predicted by the Lambertian model. Since the function R_{res} is defined on the sphere and every input image contributes one sample of R_{res}, residual modeling can be thought of as an instance of

[3]In our implementation, R_{base} is taken to be the average color at the voxel's projection in the m input images.

scattered data approximation on the sphere [35]—a rich literature on the topic exists, partly motivated by the problem of BRDF estimation [36]. Radiance residuals for a voxel can therefore be modeled using three main steps: (1) map every sample of R_{res} to its corresponding point on the sphere S^2, (2) define R_{res} for *every* point on S^2 by linearly interpolating the input samples over the entire sphere, and (3) compute a compact description of the interpolated function R_{res}.

In general, when the input samples of R_{res} are distributed arbitrarily on S^2, linear interpolation can be achieved by first performing a Delaunay triangulation [37] of the positions of input samples on S^2 and then linearly interpolating the values of R_{res} at each triangle's vertices to the triangle interior. Similarly, a compact description for the interpolated R_{res} can be derived by representing the function in terms of spherical wavelet coefficients [38]. Rather than treating the problem in its full generality, our implementation focused on the special case where all views are taken along a single-axis rotation of the object. This allows us to reduce the dimensionality of the approximation problem and to considerably simplify computations involved in these steps.

More specifically, single-axis object rotations ensure that all residual samples can be thought of as lying along a great circle of S^2. Interpolation along this circle is achieved by interpolating samples that are adjacent along the circle. To establish a value for R_{res} at points away from this circle, we define it according to the equation

$$R_{res}(\theta, \phi) = R_{res}(\theta)\cos(\phi). \tag{8}$$

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. From a computational point of view, it allows the function R_{res} to be completely described by its one-dimensional profile along a single great circle.

We use a simple Haar wavelet compression scheme [39] to represent R_{res}'s 1D profile along this circle:

$$R_{res}(x) = \sum_{k=1}^{K} w_k W_k\left(\frac{x}{2\pi}\right), \quad x \in [0, 2\pi) \tag{9}$$

where w_k are the stored wavelet coefficients that constitute R_{res}'s representation and $W_k(x), k = 1, \ldots, K$ are the K-first normalized Haar basis functions, $\phi_0^0, \psi_0^0, \psi_0^1, \psi_1^1, \ldots$ defined by

$$\phi_i^j(x) = \sqrt{2^j}\phi(2^j x - i), \quad i = 0, \ldots, 2^j - 1 \tag{10}$$

where

$$\phi(x) = \begin{cases} 1 & \text{for } 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

and

$$\psi_i^j(x) = \sqrt{2^j}\psi(2^j x - i), \quad i = 0, \ldots, 2^j - 1 \tag{12}$$

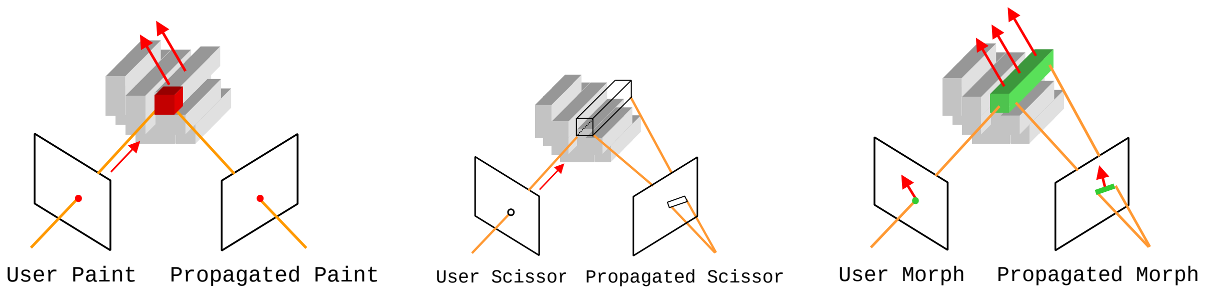
where

$$\psi(x) = \begin{cases} 1 & \text{for } 0 \leq x < 1/2 \\ -1 & \text{for } 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Besides its efficient implementation, a key feature of this scheme is that it allows interactive control of the level of detail of R_{res}'s representation: A user can specify either a maximum per-voxel data storage limit (i.e., the value of K), or a maximum allowable difference between the voxels' appearance in an input image and the appearance predicted by their combined Lambertian+residual model (i.e., K is automatically chosen for each voxel to achieve the user-provided error tolerance).

4 Implementation

This section 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.



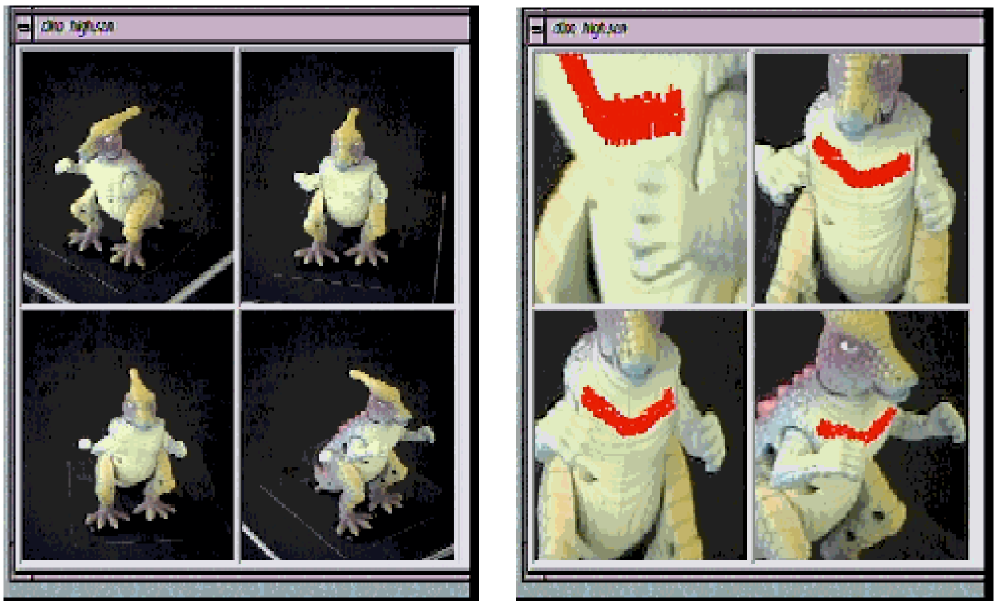
(a) (b) (c)

Figure 5: Implementation of plenoptic image painting, scissoring, and morphing operations. (a) Changing the color of a pixel has the effect of repainting the corresponding voxel, causing all other images to be modified. (b) A scissoring operation masks out a pixel in one image, causing the deletion of all voxels that project to that pixel. (c) An image warp or "morph" distorts the voxel grid parallel to the image plane, causing a shape change in all other 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 (Figure 5(a)).

Plenoptic painting can be performed in real-time by precomputing the mapping between pixels in each image and voxels in the plenoptic decomposition. In particular, we need to compute (1) for each voxel which pixels it projects to in each image, and (2) for each pixel, what is the closest voxel that projects to that pixel. Both tasks are accomplished using an *item buffer* technique[40], in which each voxel is assigned a unique pseudo-color and rendered to each viewpoint, using Z-buffering to account for visibility. The resulting images provide a direct two-way mapping between voxels and pixels in every view. Our implementation used this technique to allow a user to paint in several images simultaneously



(a) (b)

Figure 6: Screen-shots from an interactive plenoptic painting tool. (a) Four of the 21 input dinosaur photographs are selected for viewing by the user. (b) The user selects the upper-right image for painting, after having interactively scaled all four photographs. The painting operation is propagated in real time (approximately 30Hz) to the three other views of the dinosaur.

via real-time propagation of pixel edits (Figure 6).

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 (Figure 5(b)). The latter method was used to generate the images in Figure 2 (c) and (f).

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. This procedure consists of the following steps. Let I_e denote the edited image and I_p an image which is to be modified automatically as a result propagating the scissoring operation.

1. The pixels to remove are specified in I_e using a region selection tool [1, 25].
2. The corresponding voxels are removed from the plenoptic decomposition, and the corresponding pixels are masked out in I_p. Note that these correspondence can be computed in real time, using the method described in Section 4.1.
3. The new plenoptic model is rendered to the viewpoint of I_p to produce image $I_{p'}$, as described in Section 5. The masked pixels in I_p are replaced with the pixels at the same positions in $I_{p'}$.

Importantly, the original images are modified only in regions that are affected by the scissoring operation.

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 by the image pixel displacements, the motion of voxels along rays is not. Our implementation of plenoptic image morphing fixed this variable by constraining all voxels to move parallel to the image plane and used Beier and Neely's method [27] to generate image warps (Figure 7). Morph propagation was achieved by using the projected voxel displacements to define image warps in new views (Figure 5(c)). Voxels that become unoccluded as a result of the morph are rendered directly, in the same manner as described for the scissoring operation.

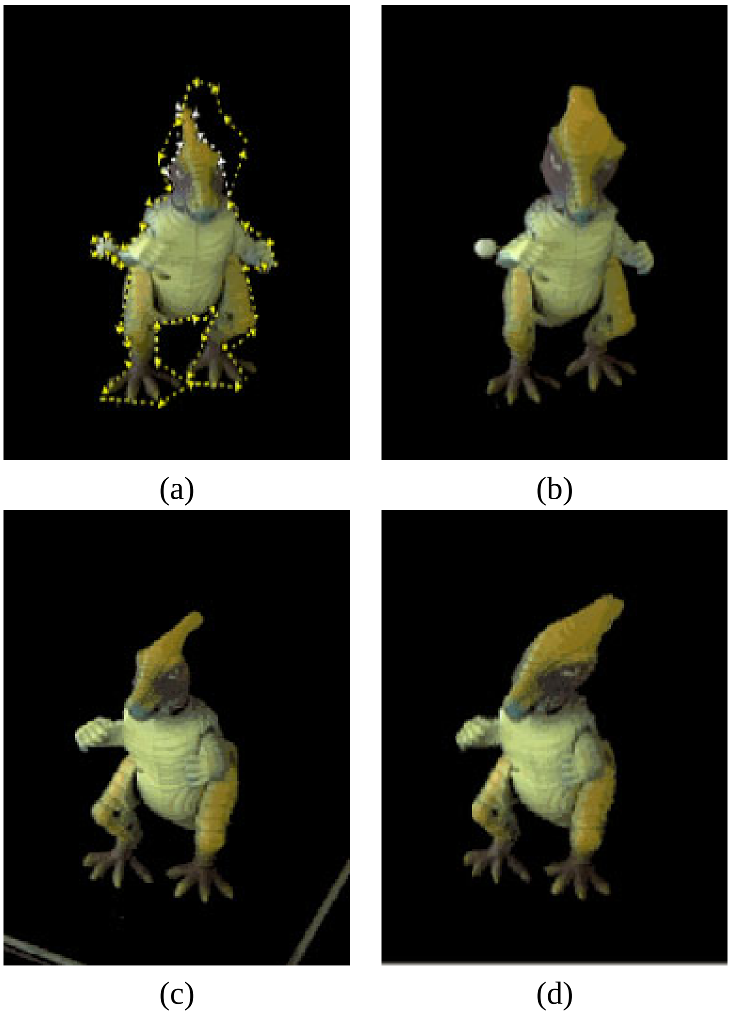


Figure 7: Screen-shots from an interactive plenoptic morphing tool. (a) One of the 21 input dinosaur photographs is selected for editing by the user. The mouse is then used to interactively draw two curves in the image: (1) a *source* curve (white dotted line around the dinosaur's silhouette), and (2) a *target* curve (yellow dotted line). A 2D image warp that maps the source curve onto the target curve is then computed automatically. The resulting warp is shown in (b). (c) A second image is selected for viewing by the user and warped automatically by the system in (d) by propagating the user-specified warp in (a).

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 [31]. 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.

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 (Eq. (1)) for each voxel V and (2) projecting the colored voxels into the image, as described next.

5.1 Specifying Illumination

Each voxel's radiance function is parameterized by a rotation matrix M which encodes the orientation of the scene with respect to the camera. Note that the light source positions are unknown, but assumed fixed with respect to the known camera frame. Consequently a change in light source direction may be simulated by evaluating Eq. (1) with the desired rotation matrix M. A variety of natural user interfaces exist for specifying rotations via direct manipulation, see for example [40].

5.2 Rendering Voxel Models

Once the illumination is set and voxel radiances are determined, a new image is generated by assigning colors (radiances) to the voxels and projecting the model to the desired viewpoint. This projection can be computed very efficiently using voxel-splatting techniques [41, 42]. These techniques approximate each voxel's projection in an image by a 2D mask and use *Z-buffering* or depth-ordering techniques [3] to account for visibility.

Figures 8(a)-(f) compare renderings of the plenoptic decomposition recovered from 21 images of a 360 degree rotation of a dinosaur toy with two different radiance models. These images show that fine

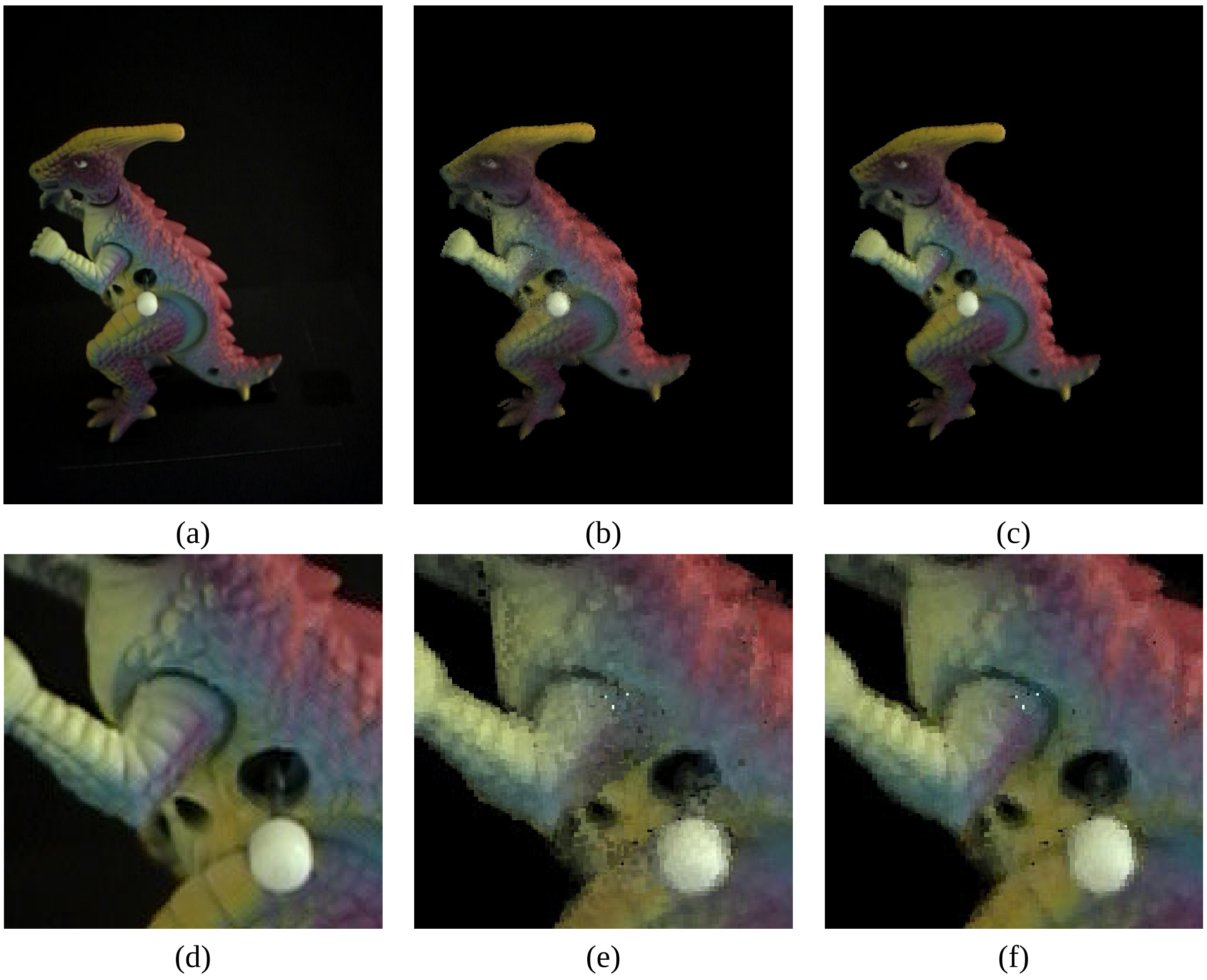


Figure 8: Reconstructing the plenoptic function of a toy dinosaur. (a) shows one of the 21 original dinosaur images. (b) and (c) show renderings of the dinosaur for the same viewpoint as the image in (a): the image in (b) was generated using the Lambertian model and (c) includes a residual model for the radiance that uses a maximum of 20 wavelet coefficients for each voxel. Enlargements (d)-(f) show detail of the dinosaur's mid-section in images (a)-(c), respectively.

image detail can be restored by modeling radiance residuals, as seen in the blowup. Figures 9(a)-(d) show the extrapolation capabilities of the system using the recovered plenoptic decomposition from 21 views of a real flower. The images demonstrate the ability of the algorithm to synthesize both changes in camera position and in light source positions.

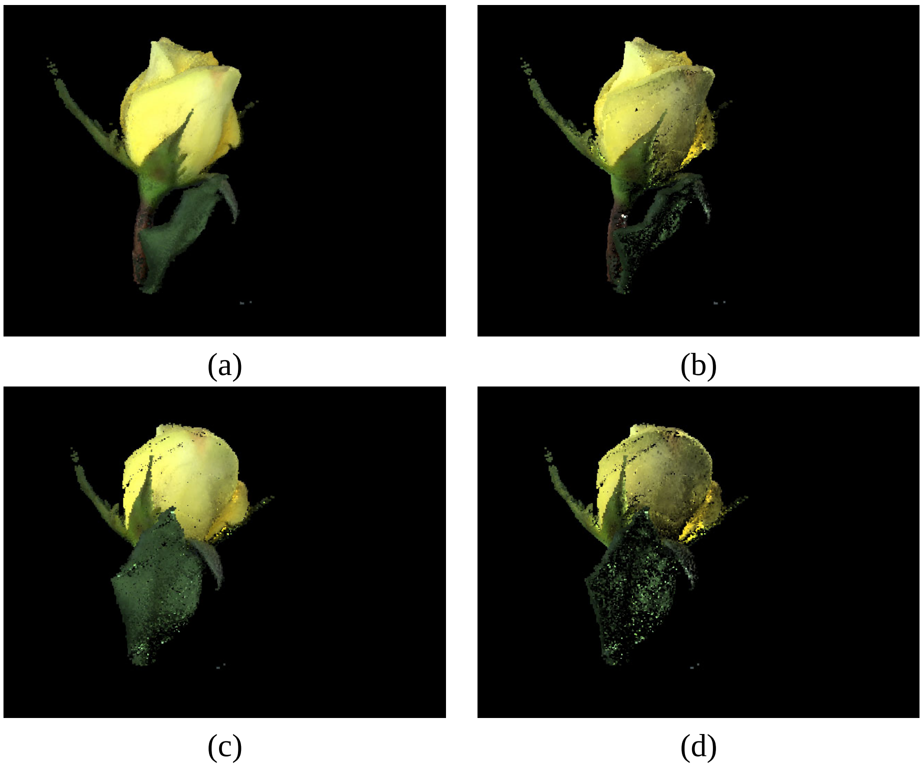


Figure 9: View and illumination synthesis applied to photographs of a rose. (a) shows one of the 21 original rose images. (b)-(d) illustrate the extrapolation capabilities of our system—the views were synthetically generated and did not have counterparts in the input sequence: in (b) the rose is viewed from the position of (a) but its illumination has been modified interactively; (c) shows a view of the rose from below, and (d) shows the same view but under illumination conditions identical to (b). These lighting effects were produced by evaluating the voxels' radiance function for viewing parameters other than those used to project the voxels.

6 Conclusions

Plenoptic image editing puts forth a new class of image editing operations that allow pixel edits in one image to be automatically propagated to all possible views of an object, in a physically-consistent manner. 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-based reconstruction to obtain 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, this method is susceptible to shape errors in low-contrast, untextured regions of the scene. We are currently studying ways to implement plenoptic editing operations when accurate correspondence information cannot be recovered from the input views [43]. We are also investigating (1) 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, and (2) ways to incorporate plenoptic edits that allow both removal and addition of pixels in the input images.

Acknowledgements The support of the National Science Foundation under Grant Nos. CDA-9503996 and IIS-9875628 is greatfully acknowledged.

References

[1] Adobe Systems, Inc., *Photoshop^{TM} 5.0*, 1998. Computer Program.

[2] 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.

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

[4] S. E. Chen and L. Williams, “View interpolation for image synthesis,” in *Proc. SIGGRAPH 93*, pp. 279–288, 1993.

[5] S. Laveau and O. Faugeras, “3-D scene representation as a collection of images,” in *Proc. Int. Conf. on Pattern Recognition*, pp. 689–691, 1994.

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

[7] M. Levoy and P. Hanrahan, “Light field rendering,” in *Proc. SIGGRAPH ’96*, pp. 31–42, 1996.

[8] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, “The lumigraph,” in *Proc. SIGGRAPH’96*, pp. 43–54, 1996.

[9] 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.

[10] S. Avidan and A. Shashua, “Novel view synthesis in tensor space,” in *Proc. Computer Vision and Pattern Recognition Conf.*, pp. 1034–1040, 1997.

[11] P. J. Narayanan, P. W. Rander, and T. Kanade, “Constructing virtual worlds using dense stereo,” in *Proc. Int. Conf. on Computer Vision*, pp. 3–10, 1998.

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

[13] P. N. Belhumeur and D. J. Kriegman, “What is the set of images of an object under all possible lighting conditions,” in *Proc. Computer Vision and Pattern Recognition*, pp. 270–277, 1996.

[14] R. C. Bolles, H. H. Baker, and D. H. Marimont, “Epipolar-plane image analysis: An approach to determining structure from motion,” *Int. J. Computer Vision*, vol. 1, pp. 7–55, 1987.

[15] A. Katayama, K. Tanaka, T. Oshino, and H. Tamura, “A viewpoint dependent stereoscopic display using interpolation of multi-viewpoint images,” in *Proc. SPIE*, vol. 2409A, pp. 21–30, 1995.

[16] S. M. Seitz and C. R. Dyer, “Physically-valid view synthesis by image interpolation,” in *Proc. IEEE Workshop on Representations of Visual Scenes*, pp. 18–25, 1995.

[17] R. T. Collins, “A space-sweep approach to true multi-image matching,” in *Proc. Computer Vision and Pattern Recognition Conf.*, pp. 358–363, 1996.

[18] A. W. Fitzgibbon and A. Zisserman, “Automatic camera recovery for closed or open image sequences,” in *Proc. 5th European Conf. on Computer Vision*, pp. 311–326, 1998.

[19] R. Szeliski, “Rapid octree construction from image sequences,” *CVGIP: Image Understanding*, vol. 58, no. 1, pp. 23–32, 1993.

[20] R. Cipolla and A. Blake, “Surface shape from the deformation of apparent contours,” *Int. J. Computer Vision*, vol. 9, no. 2, pp. 83–112, 1992.

[21] S. Moezzi, A. Katkere, D. Y. Kuramura, and R. Jain, “Reality modeling and visualization from multiple video sequences,” *IEEE Computer Graphics and Applications*, vol. 16, no. 6, pp. 58–63, 1996.

[22] K. N. Kutulakos, “Shape from the light field boundary,” in *Proc. Computer Vision and Pattern Recognition*, pp. 53–59, 1997.

[23] P. Hanrahan and P. Haeberli, “Direct WYSIWYG painting and texturing on 3d shapes,” in *Proc. SIGGRAPH’90*, pp. 215–223, 1990.

[24] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes, “SKETCH: an interface for sketching 3D scenes,” in *Proc. SIGGRAPH’96*, pp. 163–170, 1996.

[25] E. N. Mortensen and W. A. Barrett, “Intelligent scissors for image composition,” in *Proc. SIGGRAPH’95*, pp. 191–208, 1995.

[26] M. Gleicher, “Image snapping,” in *Proc. SIGGRAPH’95*, pp. 183–190, 1995.

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

[28] D. Wilkes and J. K. Tsotsos, “Active object recognition,” in *Proc. Computer Vision and Pattern Recognition*, pp. 136–141, 1992.

[29] 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.

[30] S. Benton, “Survey of holographic stereograms,” in *Processing and Display of Three-Dimensional Data*, Proc. SPIE, 1983.

[31] S. M. Seitz and C. R. Dyer, “Photorealistic scene reconstruction by voxel coloring,” *Int. J. Computer Vision*, vol. 35, no. 2, 1999.

[32] K. N. Kutulakos and S. M. Seitz, “A theory of shape by space carving,” *Int. J. of Computer Vision*, vol. 38, no. 3, pp. 197–216, 2000.

[33] 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, University of British Columbia, Laboratory for Computational Intelligence, August 1991.

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

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

[36] 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.

[37] F. P. Preparata and M. I. Shamos, *Computational Geometry*. Springer-Verlag, 1985.

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

[39] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, 1996.

[40] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics Principles and Practice*. Addison-Wesley Publishing Co., 1990.

[41] B. Lichtenbelt, R. Crane, and S. Naqvi, *Introduction to Volume Rendering*. Prentice-Hall, 1998.

[42] J. W. Shade, S. J. Gortler, L. wei He, and R. Szeliski, "Layered depth images," in *Proc. SIGGRAPH 98*, pp. 231–242, 1998.

[43] K. N. Kutulakos, "Approximate N-View stereo," in *Proc. European Conf. on Computer Vision*, pp. 67–83, 2000.