To appear in the Symposium on Interactive 3D Graphics and Games (I3D) 2005 conference proceedings

Interactive Deformation of Light Fields

Billy Chen* Stanford University
Eyal Ofek† Microsoft Research Asia
Heung-Yeung Shum† Microsoft Research Asia
Marc Levoy* Stanford University

Figure 1: Light field deformation enables an animator to interactively deform photo-realistic objects. The left figure shows an image of a light field of a toy Terra Cotta Warrior. The middle image shows a view of the same light field after applying a deformation, in this case, a twist to the left. Notice that his feet remain fixed and his right ear now becomes visible. The right image shows the warrior turning to his right.

Abstract

We present a software pipeline that enables an animator to deform light fields. The pipeline can be used to deform complex objects, such as furry toys, while maintaining photo-realistic quality. Our pipeline consists of three stages. First, we split the light field into sub-light fields. To facilitate splitting of complex objects, we employ a novel technique based on projected light patterns. Second, we deform each sub-light field. To do this, we provide the animator with controls similar to volumetric free-form deformation. Third, we recombine and render each sub-light field. Our rendering technique properly handles visibility changes due to occlusion among sub-light fields. To ensure consistent illumination of objects after they have been deformed, our light fields are captured with the light source fixed to the camera, rather than being fixed to the object. We demonstrate our deformation pipeline using synthetic and photographically acquired light fields. Potential applications include animation, interior design, and interactive gaming.

Keywords: light fields, deformation, image-based rendering, 3D photography

*e-mail: {billyc|levoy}@graphics.stanford.edu
†e-mail: {eyalofek|hshum}@microsoft.com

1 Introduction

A light field enables photo-realistic rendering of objects and scenes without knowing their geometry [Levoy and Hanrahan 1996]. Since they operate on captured photographs, light fields remove the need to explicitly model geometry, lighting, or surface reflectance properties. Novel views of interesting objects are found in [Matusik et al. 2002], where they render light fields of furry teddy bears, toy angels, and plants; such objects have a very complex geometry and complex reflectance models.

The ability to deform an object has many uses, including modeling and animation. In cartoon animation, "squash" and "stretch" deformations are widely used to enhance motion and character [Johnston and Thomas 1995]. In modeling, deformation is often used to simulate flexible materials. Unfortunately, deforming a three-dimensional object typically requires having a representation of the object's geometry. Light fields, on the other hand, have no geometry, or perhaps just a simple proxy [Gortler et al. 1996]. Therefore, it is not immediately clear how to apply deformations to light fields.

In this paper we describe such a technique for deforming light fields. By doing so, an animator can leverage both the photorealism of image-based rendering and the expressive power of deformation. Figure 1 illustrates a deformation on a light field of a Terra Cotta soldier. We ensure consistent illumination of the soldier as it deforms by capturing a light field of it in which the illumination is fixed near the camera as it moves. We call this image-based representation a *coaxial light field*. In Section 3 we define this representation in more detail, and we show that it keeps the illumination of objects "consistent" during deformations.

Our method for deforming a light field is performed in three stages: splitting the light field into sub-light fields, deforming each one, and rendering them together.

First, the light field is split into sub-light fields. This is done by partitioning 3D space into multiple subvolumes. With each 3D subvolume we associate a subset of the light field, namely those rays that intersect objects lying inside that subvolume. Figure 2 illustrates this notion. In this paper, we assume that subvolumes are rectangular parallelepipeds, and we henceforth refer to them as "deformation boxes" or simply "boxes"[1]. To associate rays of the light field with particular deformation boxes, one can use a variety of techniques. The technique we present in this paper uses video projectors. Specifically, during image capture we illuminate that portion of the object lying in each deformation box with a different color. By analyzing the colors of pixels in each view of the light field, we can associate the ray corresponding to that pixel with the appropriate subvolume and hence the appropriate sub-light field.

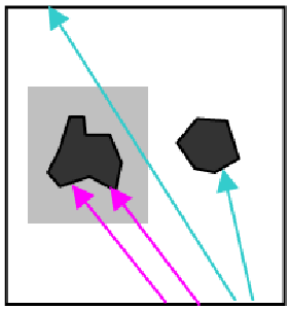

Figure 2: An illustration of a sub-light field. In this top-down view, two objects are shown in black. The subvolume is drawn in light gray. Rays that strike the object in that subvolume, and are hence in its sub-light field, are drawn in pink; rays that either do not intersect the subvolume or that do not strike the object are drawn in blue. These rays are not included in the sub-light field.

Second, a deformation is applied to each sub-light field. The animator specifies the deformation by moving the vertices of the corresponding box. We then apply a 3D warp that maps the undeformed box to a deformed one, transforming the rays associated with that box. Figure 3 illustrates how a deformation affects the rays in the sub-light field. The variant of trilinear warps we employ in this paper maps straight lines to straight lines, thereby producing straight viewing rays. This allows us to extract the rays we need from the light field we have captured.

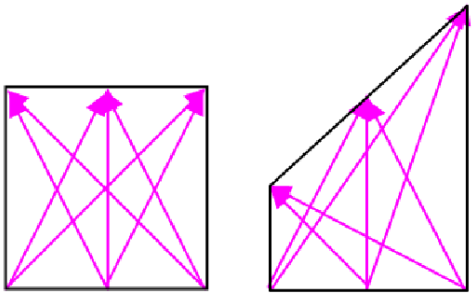

Figure 3: The left image shows an undeformed box (in black) and several rays (pink arrows). In the right image, the animator manipulates the deformation box, thus causing a 3D warp on the rays.

Finally, the deformed sub-light fields are joined together and rendered using a technique that preserves the occlusion ordering of the subvolumes.

[1]Our technique would also work using other subvolume shapes, but we found boxes to be simple and effective for deformation.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the coaxial light field and discusses why it is useful for deformation. Section 4 presents techniques for splitting a light field into sub-light fields. Section 5 illustrates how to define a light field deformation. Section 6 shows how to combine and render multiple deformed light fields using hardware-accelerated texture mapping. Section 7 shows our implementation and results. Section 8 discusses the limitations of our technique and future work.

2 Related work

Light fields [Levoy and Hanrahan 1996; Gortler et al. 1996] are a popular alternative to geometric representations of scenes, allowing photorealistic flythroughs of complex scenes at interactive rates. A survey of light fields and other image-based representations is given in [Shum and Kang 2000].

Aside from rendering, relatively little research has been done on light fields; in particular on editing them. Notable exceptions are [Seitz and Kutulakos 1998; Rangaswamy 1998] which allow the user to perform image editing operations such as painting, cropping and morphing on a set of images. The edits in a single image are propagated to other images via an underlying volumetric model. [Oh et al. 2001] allows cloning and texture-illuminance decoupling in a layered-depth image representation of the scene.

The editing technique most similar to ours is [Zhang et al. 2002], which permits morphing of a source light field into a target one. Corresponding 3D features are specified in the source and target light fields by the user. These corresponding points are then used to drive a warping of the two light fields, followed by a blending between them. In our method we also warp the light field, but we define deformation as a 3D spatial transformation. In other words, we warp the volume that contains the object, as opposed to warping the specified feature points on the object. Although it does not directly support morphing between two light fields, we believe our formulation makes it easier for users to animate a single light field.

So far, rendering these light fields have used little or no geometry. If we relax this restriction and provide a geometric proxy of the scene, then deformation can be realized by warping the proxy and preserving the surface reflectance properties during this process. Weyrich et al., presented a system which allows for the relighting and deformation of surface reflectance fields [Weyrich et al. 2004]. Given an "impostor" geometry on which a surface reflectance field is defined, they deform this representation and show that it approximately preserves the material properties of the object. Our technique applies a warp on rays, which facilitates interactive deformation independent of the complexity of the geometry. However, our implementation trades off the accuracy in the appearance of the deformed object for interactive rendering rates.

Finally, deformations on volumes (e.g. 3D points) have also been explored in the computer graphics and simulation communities. However, no one to date has applied these deformation techniques to light fields, i.e. to collections of 4D rays. Our work is inspired by the work of Alan Barr [1984], who observed that a deformed object can be rendered by tracing warped rays through the undeformed space containing the object. Kurzion and Yagel also extended the original idea by defining "ray deflectors" to deform rays before they entered a rendering system [Kurzion and Yagel 1996]. Our technique uses deformation boxes, a technique similar to volumetric free-form deformation, which provides the animator with an intuitive way to specify the ray warp.

Before we discuss the deformation process, we first show why using a coaxial light field helps produce more consistent lighting during

deformation.

3 Coaxial light fields

[Levoy and Hanrahan 1996] defines the 4D light field as radiance along rays as a function of position and direction in a scene under fixed lighting. Their definitions permit construction of new views of an object, but its illumination cannot be changed. By contrast, [Debevec et al. 2000] defines the 4D reflectance field as radiance along a particular 2D set of rays, i.e. a fixed view of the world, as a function of (2D) direction to the light source. Their definition permits the relighting of an object, but the observer viewpoint cannot be changed. If one could capture an object under both changing viewpoint and changing illumination, one would have an 8D function (recently captured by [Goesele et al. 2004]). The light fields of [Levoy and Hanrahan 1996] and [Debevec2000] are 4D slices of this function.

In this paper, we define a different 4D slice, which we call the "coaxial light field." Specifically, we capture different views of an object, but with the light source fixed to the camera as it moves. In fact, we assume that the camera rays and illumination rays coincide. Since perfectly coaxial viewing and illumination is difficult to achieve in practice, we merely place our light source as close to our camera as we can. As an alternative, a ring light source could also be used.

Let us consider what happens to the illumination on a diffuse object when rendering a deformed coaxial light field. Figures 4 and 5 illustrate the simulated capture and rendering of a deforming checkerboard. In the first figure, coaxial illumination is used during capture, i.e. the light source is fixed to the camera. In the second figure, the light source is fixed to the scene. As the figures show, only the first case generates a correct rendering of the deformed checkerboard.

Thus, the advantage of coaxial viewing and illumination is that it ensures the correct appearance of objects under deformation, even though no geometry has been captured. However, this technique has several limitations. First, the object must be diffuse; specular highlights will look reasonable when deformed, but they will not be correct. Second, perfectly coaxial lighting contains no shadows. This makes them look somewhat flat. In practice, our light source is placed slightly to the side of our camera, thereby introducing some shadowing, at the expense of slightly less consistency in the rendering.

Interestingly, if the light source is placed farther from the camera, and we assume that the camera orbits around the center of the object, then the rendering becomes inconsistent only for parts of the object that are far from this center. Moreover, it become inconsistent only gradually. We are currently experimenting with this extension. If the artifacts it produces are not too serious, then it gives the animator more control over the placement of lights, which in turn allows the creative use of shadows.

4 Splitting the light field

Now we begin our discussion of the deformation process by describing how we split a light field. The purpose of splitting a light field is so that each sub-light field can undergo a different deformation. We define a sub-light field as a collection of two objects: a bundle of rays and an associated deformation box. Our goal is to associate a deformation box for each ray bundle. A ray bundle consists of the rays that hit object points that undergo the same transformation. For example, in Figure 6 one ray bundle (shown in yellow) is the set of all rays that hit a point on the fish head. In another view of the fish, we would see other rays in this same bundle that also hit the head. To store this association, we augment each deformation box with an index, and we label each associated ray with that index.

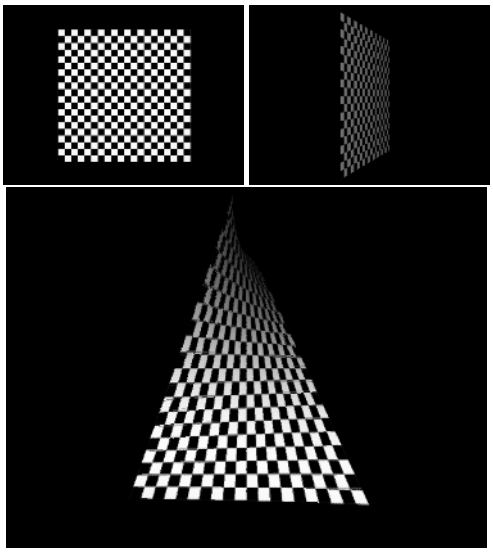

Figure 4: The top two images are from a simulated coaxial light field. The lighting is a point light source placed near the camera. As the camera moves to an oblique position, the plane becomes dark because irradiance falls off with the angle between the plane's normal and the illumination direction. The plane is assumed to reflect light diffusely. The bottom image shows the checkerboard after application of a twisting deformation in which the top edge is rotated along the vertical axis while the bottom edge stays fixed. From the checkerboard's bottom edge to its top edge, it becomes darker. Although this image was made by rendering a deformed light field, this change in intensity correctly depicts what the checkerboard would look like if it were twisted in the original scene. The artifacts are due to under-sampling in the light field and are not related to the correct lighting on the deformed checkerboard.

For light fields from synthetic data, creating ray bundles is straightforward, since the object geometry is known. First, we split the triangle mesh algorithmically into parts, where each part undergoes a different deformation. Then, in 3D, we place a deformation box over each sub-mesh. The ray bundle which belongs to a particular box consists of all the rays that are incident to object points within that box. To compute these ray bundles, we color each submesh with a different color and capture a light field of the object. In this way, the color of each ray denotes the bundle to which this ray belongs. Figure 6 shows one view of the colored mesh and the associated deformation boxes.

For light fields from captured data, there are several techniques that could be employed for specifying the ray bundles. In the simplest case, when the cameras are arranged in a circle and properly aligned with respect to an object, scan lines in the images can be used to segment the light field into ray bundles. For example, in Figure 1 all pixels (e.g. rays) in all images above the scan line containing the statue's neck could be bundled together. This would allow us to rotate only the statue's head while applying a different deformation to its body. In Figure 1, we actually use a single deformation box over the entire statue. Using camera scan lines to split the light field limits how we can acquire an object, which objects can be deformed, and the space of animations we can apply to them.

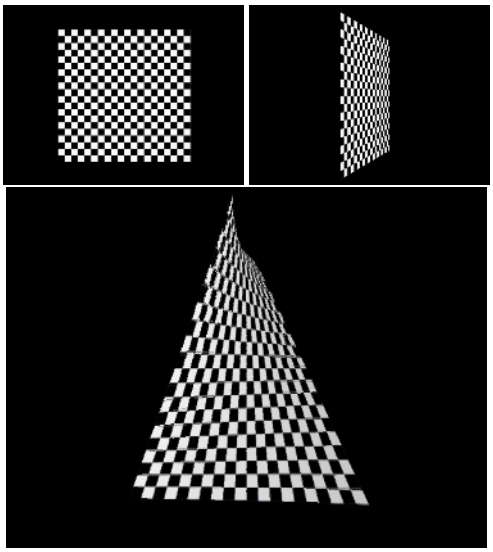

Figure 5: As in Figure 4, the top two images are images from a simulated light field. In this case, the lighting is a point source fixed relative to the plane, while the camera revolves around it. Since the checkerboard is diffuse, its apparent brightness remains unchanged as the camera moves. The bottom image again shows the checkerboard after application of a twisting deformation. However, this time the checkerboard remains uniformly white. This is incorrect for a twisted plane illuminated by a point light source.

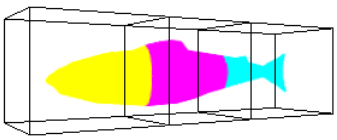

Figure 6: Splitting a synthetic light field. The fish is split into three sub-light fields, each bounded by a deformation box. In this view, the rays belonging to each sub-light field are color coded. The three parts of the fish can now undergo independent warps.

4.1 Projector-based segmentation of light fields

To alleviate these restrictions, one can use video projectors to designate the ray bundles. We show that this technique also enables us to render more complex motions. First we describe our acquisition procedure for the teddy bear shown in Figure 14. At the end of this section we discuss advantages and limitations of this technique.

To segment our light field of the teddy bear we actually capture three light fields: 1) a coaxially illuminated one used for rendering, 2) a light field under color-coded illumination from multiple projectors, and 3) one under floodlit illumination from the same projectors. The second and third light fields are used to designate ray bundles for deformation. Figure 8 illustrates the acquisition setup using the Stanford Spherical Gantry [Levoy 2004]. We use two projectors to throw colors onto the teddy bear. One projector illuminates the teddy bear's front, and the other his back. The colors denote ray bundles that we will deform independently. Figure 7 illustrates this coloring. The colors are chosen such that their distance in color space as seen by the projector-camera system is maximal. In particular, the color difference between two different areas should be larger than the color variance due to noise.

The image that each projector emits is created by hand. Specifically, a person sits at a computer that has two video outputs. The video outputs are exact clones of each other. One output is shown on a standard CRT monitor. The other output is displayed through a projector, aimed at the teddy bear. Using a drawing program displayed on the CRT monitor, the user paints colored regions which are displayed live onto the teddy bear, through the projector. This technique emulates the synthetic case where sub-meshes of an object are colored in order to compute the ray bundles. We use two projectors aimed at the teddy bear's front and back respectively. Drawing the images that the projector emits is relatively easy since the projected colors are not used for recovery of the model, but merely to split the light field into sub-light fields. After painting these colored regions (which are projected onto the teddy bear), we then capture a light field under this new illumination.

In addition to acquiring this colored light field, another one is acquired under floodlit illumination from the projectors. These images are used to normalize the data from the colored light field, thus reducing effects from varying object albedo or non-uniformity of the illumination brightness. Normalization is computed by the following equation:

$$B' = 256 * B/W \tag{1}$$

where W is the floodlit image (8 bits per channel), B is the colored image and B' is the normalized color image. Figure 7 shows image B' for one camera's view of the teddy bear light field.

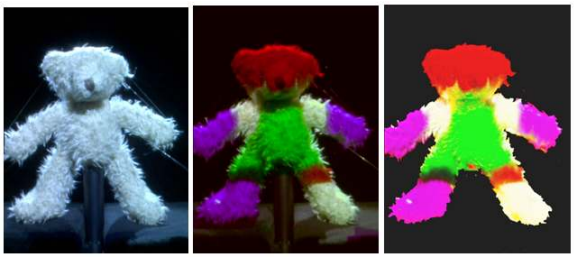

Figure 7: Segmenting a teddy bear light field by using projector illumination. The left image is the bear under floodlit illumination. The middle image shows the same bear when projected with color masks denoting ray bundles. The colors designate ray bundles for the head, torso, arms, legs and joints. Each color denotes the sub-light field to which that pixel belongs. On the right is the same view after normalization. These illumination conditions are captured for each camera position in the light field.

Finally, deformation boxes are created to surround colored regions of the object in the light field . We first construct boxes in 3D using a modeling program like 3D Studio Max. Then, we project the boxes onto each camera's view and verify that these projections encompass the pixels that have the associated color. For the teddy bear light field, we only project the deformation boxes onto a few views, for example, the front, back and side views of the bear.

In our approach, we specified ray bundles using projectors, then associated a deformation box to each bundle. This forced us to decide, at acquisition time, on the number and nature of the independent deformations that would later be applied. An alternative approach would be to first specify the deformation boxes and then associate ray bundles to each box. One way to do this would be to specify a box in 3D and project it onto each view in the captured light field. In each such view, the user interactively paints those pixels (e.g. rays) which both hit the object and lie in the projection of the appropriate deformation box. The user would have to paint pixels in every view of the light field, which is a time consuming process. However, this could be facilitated by a geometric proxy that would be used to propagate painted pixels to other views.

No matter how these ray-to-box associations are derived, it is desirable that only rays striking object points lying inside a deformation box are associated with that box. However, when this condition is violated, rays can either be warped when they shouldn't be, or vice versa. We handle the former problem by only warping those rays that pass through the deformation box (see Section 5). In this way even if rays are mislabeled (e.g. they hit object points lying outside of the deformation box), they are not rendered.

While projectors have proven useful for the objects we deformed, using them does have its limitations. First, this technique would be difficult to apply for objects of high reflectivity, transparency or very low albedo. Second, self shadows can make segmentation harder. Third, sometimes projectors are not appropriate tools for splitting the light field at all, i.e. for segmenting scenes into fronto-parallel sub-light fields. In this case, one could extend the focused imaging techniques described in [Levoy et al. 2004]. By focusing the projectors' illumination on the object's surface, we can also alleviate problems due to self-shadowing. Alternatively, by combining "shape from focus" or "shape from stereo" operators with focused imaging, we could segment the light field into fronto-parallel sub-light fields. An application of such a segmentation is animating the depth layers of a scene containing a field of wheat or grass, thus simulating the effect of wind.

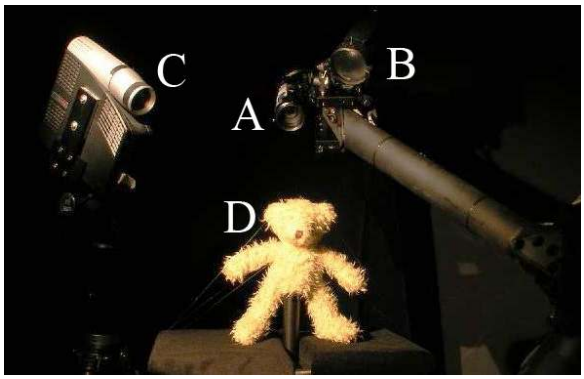

Figure 8: Our acquisition setup for capturing a cylindrical light field (as defined in Section 6.2) of an object. Using the Stanford Spherical Gantry, the camera (A) rotates in a circle in the horizontal plane. Two lights are placed close to the camera to create near-coaxial lighting. One light (B) is shown in the above picture. Two projectors (C, only one shown) are placed above and outside of the gantry and illuminate the object (D).

5 Deformation

After splitting the light field into sub-light fields, each one is deformed. A deformation is a function $D : \Re^3 \rightarrow \Re^3$. Since the geometry of the sub-light field is unknown, we must directly transform the rays. By warping the rays, the visual effect is as if we had deformed the object. We represent a ray in the light field by two points on its path. The deformation function warps these two points and produces a new ray that goes through them. To specify the deformation, we use a variant of free-form deformation [Sederberg and Parry 1986], controlled by the vertices of a box.

Mathematically, we define a deformation box C as a set of eight 3D points. The animator can move any subset of C to form C_w, a set of eight warped points. We assume the animator does not move points to form self-intersecting polytopes. The deformation D is thus a 3D function mapping C to C_w. While there are many ways to define such a deformation, any such formulation should satisfy the following three criteria:

1. D must map C to C_w
2. D must maintain C^0 continuity across deformation boxes sharing adjacent faces
3. straight lines should be preserved

The first criterion ensures that the warp adheres to the animator's decisions. The second criterion guarantees that when an object lies across adjacent deformation boxes, it is never disconnected by the deformations. The third criterion enables us to use standard light field rendering techniques to render rays. An obvious choice for the transformation that preserves linearities is the projective transformation. Unfortunately, this mapping violates the first criterion since there may not exist exact perspective transformations that map the undeformed box to the deformed one. Furthermore, since the perspective transform only approximates the deformation, the vertices of adjacent cells may no longer be coincident, causing a shared face to become disconnected.

We implemented a deformation technique using trilinear coordinates that satisfies the above criteria and is fast and easy to compute. First, we describe the deformation function, and then we analyze its behavior.

Warping with trilinear coordinates

Let us assume, without loss of generality, that the original deformation box is a unit cube with a corner at the origin. Then for any 3D point p, we can define it in terms of three interpolation coordinates, u, v, and w. By trilinearly interpolating across the volume, p can be described in terms of the interpolation coordinates and the 8 vertices of the cube:

$$\begin{aligned} p = {} & (1-u)(1-v)(1-w)c_1 + (u)(1-v)(1-w)c_2 + \\ & (1-u)(v)(1-w)c_3 + (u)(v)(1-w)c_4 + \\ & (1-u)(1-v)(w)c_5 + (u)(1-v)(w)c_6 + \\ & (1-u)(v)(w)c_7 + (u)(v)(w)c_8 \end{aligned} \tag{2}$$

where the c_i are the vertices of the cube. Simple scaling and translation can be used for the general case of a rectilinear box. [Warren et al.] present a technique for convex polytopes which reduces to the same formulation in the rectilinear case. Any 3D point can be warped using the trilinear coordinates described above.

In general, this transformation satisfies criteria 1 and 2, but is not linear. However, we use it to warp rays by transforming the entry and exit points of a ray with respect to its associated deformation box. The new entry and exit points define a new (straight) line that represents the warped ray. This ray warp is linear and hence satisfies all the above criteria. In addition, the warp is well defined as long as the deformation box does not self intersect.

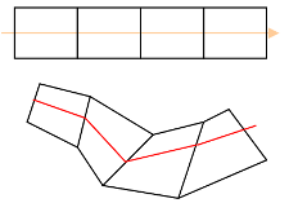

Figure 9: Warping a ray. Above, shows the undeformed boxes and a ray going through them. Below, is a conceptual illustration of the deformed boxes, the piecewise-linear deformed ray (shown in red).

6 Joining and rendering

The final stage of our framework is joining and rendering the deformed sub-light fields. For each view ray, v_i, it is partitioned into segments by the deformation boxes (a segment is the portion of the ray within a single box). Let $l_1, ..., l_n$ be the segments along v_i traversed in the forward direction. First, we warp l_1 with the inverse deformation as described in Section 6.1. Let us call this warped ray segment w_1. We then render two values, s_1 and t_1. s_1 is the value of w_1 when rendering using the colored light field (as described in Section 4). Similarly, t_1 is the value of w_1 when rendering using the coaxial light field. We describe our light field rendering algorithm in Section 6.2. If s_1 matches the label of the deformation box, then the color of v_i is t_1 and we proceed to the next view ray, v_{i+1}. Otherwise, we continue to the next segment l_2. This procedure repeats until all ray segments have been processed (in which case v_i is black), or until t_i matches the appropriate deformation box label. Figure 10 illustrates an example of using this algorithm while the following contains the associated pseudo-code.

```
RENDER_VIEW(view_pt)
1   for view_ray ← view_pt do
2       view_ray^color ← BLACK
3       l ← SEGMENT_RAY(view_ray)
4       for i ← 1...n_segments do
5           w_i ← WARP(l_i)
6           s_i ← RENDER(w_i, COLORED_LF)
7           t_i ← RENDER(w_i, COAXIAL_LF)
8           if (s_i = l_i^index) then
9               view_ray^color ← t_i
10              break
11          endif
12      endfor
13  endfor
```

6.1 Inverse ray warping

To obtain the rays, w_i, that are used in the above algorithm, we apply the inverse deformation on the ray segments l_i. The inverse deformation function is computed once for every view point we wish to synthesize. The cost of this technique is proportional to the resolution of the image. For trilinear warping, the inverse function maps the deformed box back to the original one.

In practice, evaluating the inverse is time consuming so we estimate it by forward warping many 3D samples and use interpolation to approximate the inverse point. We use a hardware-accelerated texture-mapping approach to quickly interpolate among the forward-warped 3D points. First, we divide each face of the original box into many textured triangles. The texture color codes the original coordinates of the face point on the undeformed cube. We used 64 x 64 x 2 textured triangles per face for our datasets. The triangle vertices are warped using the free-form deformation specified by the deformation box, and the warped triangles are rendered. Front-facing and back-facing faces are rendered separately and are used as a look-up table (LUT) approximation for the inverse mapping. Figure 11 illustrates the front and back-facing renderings corresponding to a given trilinear warp. Using those LUTs we recover the two 3D points of the intersection of the original ray with the undeformed cube. These points form a ray which is passed into the light field renderer.

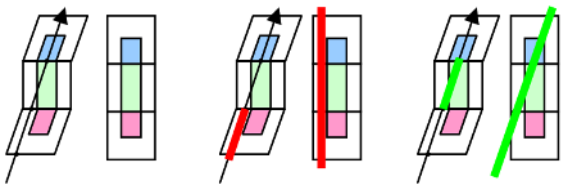

Figure 10: Rendering a view ray. The left pair of images show the boxes before and after deformation. A view ray, v_i intersects the deformation boxes. The ray is divided into segments where each one belongs to a box. The first segment, l_1 (shown in red in the middle pair of images) is warped, producing w_1. w_1 is then rendered using the colored light field, producing s_1. In this case, s_1 is black since it does not hit the object. We continue to the next segment, l_2 (shown in green in the right pair of images) and observe that s_2 matches the label for the deformation box. t_2 is then set to be the color value for the ray v_i.

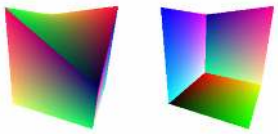

Figure 11: A backward trilinear warp is approximated by forward warping many samples and then interpolating the results. The left and right images show the front and back facing triangles, respectively. Notice that the "faces" of the warped box are no longer planar but remain smooth.

6.2 Light field rendering

We use a cylindrical light field parameterization (CLF) that is well suited for inward-looking light fields having large viewing angle and limited vertical parallax. Additionally, CLFs are easy to acquire. A CLF is parameterized by cameras lying on the surface of a cylinder of fixed radius. The viewing direction of each camera is aimed at a common point along the axis of the cylinder.

Figure 12 illustrates in flatland the rendering process for a deformed view ray, r. The process is similar to the technique used to render concentric mosaics [Shum and He 1999]. First, we compute m, the intersection point between r and the cylinder. Then we find the nearest cameras to m (shown as a and b in the figure). We then define a focal plane p which is orthogonal to r. Of the family of planes orthogonal to r, we select the one that is a distance d from the center of the cylinder. For most acquisitions, we pick d to be close to zero. Finally, we intersect r with p and project this point, x, onto the image planes of the nearest cameras to obtain a' and b'. In 3D we have four nearest cameras and we apply quadralinear interpolation on the nearest 16 rays to render a final color for r.

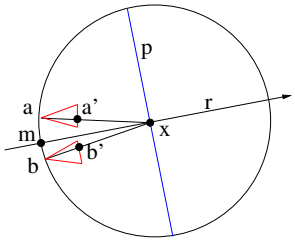

Figure 12: Rendering from a cylindrical light field.

7 Results

Figure 1 shows deformations on a cylindrical light field of a toy Terra Cotta Warrior using a single deformation box. The deformation is defined by rotating the top 4 vertices of the box. Notice that as the statue is twisted, we see the correct change in visibility. For example, the ear of the toy becomes visible as the head is turned away. Also notice that the model exhibits correct lighting after deformation. The statue is photographed under near-coaxial illumination.

In Figure 13 we animate a swimming fish by controlling three deformation boxes. The middle box is being warped, while the front box (the head) and the back box (the tail) are rotated according to the bending of the middle box. Notice that visibility is rendered correctly: the head pixels are drawn in front of the body and tail pixels.

Figure 14 shows a few frames from an animation of a furry teddy bear. Using projectors, the bear is split into several deformation boxes to allow for independent arm, leg and head motion.

8 Discussion and future work

We have presented a pipeline for deforming and rendering light fields. By splitting, deforming and joining sub-light fields, we can apply global and local deformation to photorealistic objects; this ability enables many applications. In particular, it allows an animator to use key-frame interpolation of deformations in order to produce a light field animation. Alternatively, using these deformations a modeler can customize a light field for insertion into a geometric model or another light field. Given an archive of light fields, interior designers can deform plants, chairs, etc., to rapidly prototype photorealistic scenes of rooms. Finally, game designers can capture a light field of objects (like the toy soldier), which they can animate for insertion into a video game.

There are limitations to deforming light fields. First, there is a trade-off between modeling effort and the level of animation control. With more deformation boxes, the animator can generate more complex motions, but more splitting must be done to associate the sub-light fields to the deformation boxes. Second, when splitting a light field, we assume there is a one-to-one mapping of rays to deformation boxes. This assumption is violated for objects with inter-reflections, refractions or transparency, where rays can enter and exit multiple boxes. In this case, the ray would either never undergo a deformation or would be deformed in an inconsistent manner.

Our ray-warping algorithm also has some limitations. It can twist a box, or taper it, but it can not curve it. To generate such a deformation, the animator needs to approximate the bending by further tessellating the light field into smaller boxes. An interesting extension of this work is to try to estimate the minimum necessary tessellation of a light field to perform a given transformation, with minimal of the object geometry.

When rendering multiple sub-light fields, large deformation of boxes might produce disocclusions of background boxes that were not actually being captured in the original light field. Searching for the nearest visible rays gives an approximation for the newly revealed areas, but if these areas are large, this approximation may be poor.

Finally, we introduced the coaxial light field, where the illumination is fixed near the camera as it moves. This has the limitation that shadows are reduced and that the image appears unnatural. We are currently investigating the visual effects of fixing the illumination in positions further away from the camera. This is a topic for future work.

9 Acknowledgments

We would like to thank the reviewers for their insightful comments on the paper. Thanks to Sing Bing Kang and Baining Guo for useful discussions on light field deformation. Thanks also to Elizabeth Chen, Gaurav Garg and Vaibhav Vaish for their help in preparing the paper and the video. Chen and Levoy were supported by the NSF under contract IIS-0219856-001 and DARPA under contract NBCH-1030009.

References

BARR, A. 1984. Global and local deformations of solid primitives. In *Proc. SIGGRAPH 1984*.

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H.-P., SAROKIN, W., AND SAGAR, M. 2000. Acquiring the reflectance field of a human face. In *Proc. SIGGRAPH 2000*.

GOESELE, M., LENSCH, H. P. A., LANG, J., FUCHS, C., AND SEIDEL, H.-P. 2004. Disco – acquisition of translucent objects. In *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proc. SIGGRAPH 1996*.

JOHNSTON, O., AND THOMAS, F. 1995. *The Illusion of Life: Disney Animation*. Disney Editions.

KURZION, Y., AND YAGEL, R. 1996. Continuous and discontinuous deformation using ray deflectors. In *Proc. GRAPHICON 1996*.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *Proc. SIGGRAPH 1996*.

LEVOY, M., CHEN, B., VAISH, V., HOROWITZ, M., MCDOWALL, I., AND BOLAS, M. 2004. Synthetic aperture confocal imaging. In *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*.

LEVOY, M. 2004. Stanford spherical gantry. *http://graphics.stanford.edu/projects/gantry*.

MATUSIK, W., PFISTER, H., NGAN, A., BEARDSLEY, P., ZIEGLER, R., AND MCMILLAN, L. 2002. Image-based 3d photography using opacity hulls. In *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proc. SIGGRAPH 2001*.

RANGASWAMY, S. 1998. *Interactive editing tools for image-based rendering*. Master's thesis, MIT.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometry models. In *Proc. SIGGRAPH 1986*.

SEITZ, S., AND KUTULAKOS, K. N. 1998. Plenoptic image editing. In *Proc. ICCV 1998*.

SHUM, H.-Y., AND HE, L.-W. 1999. Rendering with concentric mosaics. In *Proc. SIGGRAPH 1999*.

SHUM, H.-Y., AND KANG, S. B. 2000. A review of image-based rendering techniques. In *Proc. VCIP 2000*.

WARREN, J., SCHAEFER, S., HIRANI, A., AND DESBRUN, M. Barycentric coordinates for smooth convex sets (submitted).

WEYRICH, T., PFISTER, H., AND GROSS, M. 2004. Rendering deformable surface reflectance fields. In *IEEE Transactions on Computer Graphics and Visualization*.

ZHANG, Z., WANG, L., GUO, B., AND SHUM, H.-Y. 2002. Feature-based light field morphing. In *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*.

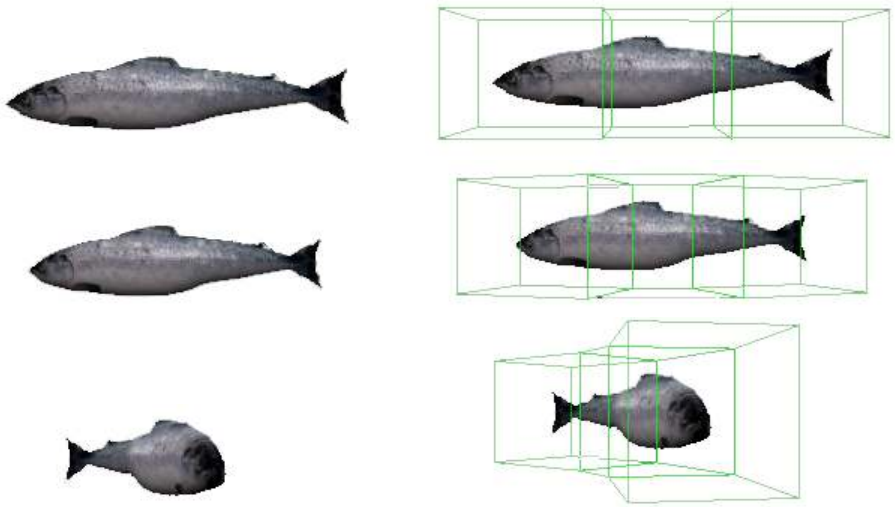

Figure 13: Deforming a fish with three independent deformations. The top-left image shows the original fish with control boxes shown in the top-right image. The middle-left image shows the deformed fish with corresponding control cubes in middle-right image. The bottom-left image shows a different view of the deformed cube. Notice that visibility changes are handled correctly by our algorithm, the fish head pixels are rendered in front of the tail pixels. This synthetic cylindrical light field was generated using 3D Studio Max

(a) (b) (c)

Figure 14: A deformation on a teddy bear. Image (a) shows a view of the original captured light field. In this case, the cameras were arranged in a ring, thus capturing a 3D subset of the full 4D light field. A total of 180 images were captured at 240 x 320 resolution. Image (b) shows a deformation in which the head, arms and legs are all bended or twisted independently. Image (c) shows the deformation boxes used to specify the motion of the teddy bear. The flash artifacts are due to specular reflection from the fish wire used to hold the bear in place.