Parallax Photography: Creating 3D Cinematic Effects from Stills

Ke Colin Zheng, University of Washington
Alex Colburn, University of Washington
Aseem Agarwala, Adobe Systems, Inc.
Maneesh Agrawala, University of California, Berkeley
David Salesin, Adobe Systems, Inc.
Brian Curless, University of Washington
Michael F. Cohen, Microsoft Research

ABSTRACT

We present an approach to convert a small portion of a light field with extracted depth information into a cinematic effect with simulated, smooth camera motion that exhibits a sense of 3D parallax. We develop a taxonomy of the cinematic conventions of these effects, distilled from observations of documentary film footage and organized by the number of subjects of interest in the scene. We present an automatic, content-aware approach to apply these cinematic conventions to an input light field. A face detector identifies subjects of interest. We then optimize for a camera path that conforms to a cinematic convention, maximizes apparent parallax, and avoids missing information in the input. We describe a GPU-accelerated, temporally coherent rendering algorithm that allows users to create more complex camera moves interactively, while experimenting with effects such as focal length, depth of field, and selective, depth-based desaturation or brightening. We evaluate and demonstrate our approach on a wide variety of scenes and present a user study that compares our 3D cinematic effects to their 2D counterparts.

Keywords: Image-Based Rendering, Photo and Image editing

Index Terms: I.3.6 [COMPUTER GRAPHICS]: Methodology and Techniques—Graphics data structures and data types;

1 INTRODUCTION

Documentary filmmakers commonly use photographs to tell a story. However, rather than placing photographs motionless on the screen, filmmakers have long used a cinematic technique called "pan & zoom," or "pan & scan," to move the camera across the images and give them more life. The earliest such effects were done manually with photos pasted on animation stands, but they are now generally created digitally. This technique, which goes by the name of the "Ken Burns effect," after the documentary filmmaker who popularized it, is now a ubiquitous feature in consumer photography software such as Apple iPhoto, Google Picasa, Photoshop Elements, and Microsoft PhotoStory.

In recent years, filmmakers have begun infusing photographs with more realism by adding depth to them, resulting in *motion parallax* between near and far parts of the scene as the camera pans over a still scene. This cinematic effect, which we will call *3D pan & scan*, is now used extensively in documentary filmmaking, as well as TV commercials and other media, and is replacing traditional 2D camera motion, because it provides a more compelling and lifelike experience.

However, creating such effects from a still photo is painstakingly difficult. The photo must be manually separated into different layers, and each layer's motion animated separately. In addition, the background layers are typically painted in by hand so that no holes appear when a foreground layer is animated away from its original position.[1]

In this paper we look at how 3D pan & scan effects can be created much more easily, albeit with a small amount of additional input. Indeed, our goal is to make creating such cinematic effects so easy that regular users can create them from their snapshots and include them in their photo slide shows with little or no effort. To that end, we propose a solution to the following problem: given a small portion of a light field [19, 6], produce a 3D pan and scan effect automatically (or semi-automatically if the user wishes to influence its content). In most of our examples, the input light field is captured with and constructed from a few photographs from a hand-held camera. We also include results from two one-shot, multi-viewpoint cameras, for which we envision our solution will be most useful. Some predict that the commodity camera of the future will have this capability [18] (perhaps beginning with the recently announced consumer stereo camera "Fuji FinePix Real3D").

The 3D pan & scan effects are generated to satisfy two main design goals:
1. The results should conform to the cinematic conventions of pan & scan effects currently used in documentary films.
2. The conventions should be applied in a fashion that respects the content and limitations of the input data.
Our approach takes as input a light field representation that contains enough information to infer depth for a small range of viewpoints. For static scenes, such light fields can be captured with a standard hand held camera [25] by determining camera pose and scene depth with computer vision algorithms, namely structure-from-motion [8] and multi-view stereo [28]. Capturing and inferring this type of information from a single shot has also received significant attention in recent years. There are now several camera designs for capturing light fields [23, 5, 20] from which scene depth can be estimated [28]. Other specialized devices, such as coded imaging systems, capture single viewpoints with depth [17].

Light fields with depth have the advantage that they can be relatively sparse and still lead to high quality renderings [6, 20] for scenes without strong view-dependent lighting effects such as mirrored surfaces. However, such sparse inputs, taken over a small spatial range of viewpoints or even a single viewpoint, present limitations: novel viewpoints must stay near the small input set, and even then some portions of the scene are not observed and thus will appear as holes in new renderings. Our approach is designed to take these limitations into account when producing 3D pan & scan effects.

Our solution processes the input to produce 3D pan & scan effects automatically or semi-automatically to satisfy our design goals. To achieve the first goal, we describe a simple taxonomy of pan & scan effects distilled from observing 22 hours of documentary films that heavily employ them. This taxonomy enables various communicative goals, such as "create an establishing shot of the entire scene", or "transition from the first subject of interest to the

[1] http://blogs.adobe.com/bobddv/2006/09/son_of_ben_kurns.html

second". Second, we describe algorithms for analyzing the scene and automatically producing camera paths and effects according to our taxonomy. Our solution then applies the appropriate effect by searching the range of viewpoints for a linear camera path that satisfies cinematic conventions while avoiding missing information and holes, and maximizing the apparent parallax in the 3D cinematic effect. Third, we describe GPU-accelerated rendering algorithms with several novel features: (1) a method to interleave pixel colors and camera source IDs to multiplex rendering and guarantee optimal use of GPU memory; (2) the first GPU-accelerated version of the soft-z [26] technique, which minimizes temporal artifacts; and (3) a GPU-accelerated inverse soft-z approach to fill small holes and gaps in the rendered output.

In the rest of this paper we describe the components of our approach, which include a taxonomy of the camera moves and other image effects found in documentary films (Section 3); techniques for automatically computing 3D pan & scan effects that follow this taxonomy (Section 4); a brief overview of our representation of a light field with depth and how we construct it from a few photographs (Section 5); and finally two rendering algorithms, both real-time (Section 6.1) and off-line (Section 6.2). We then demonstrate results for multiple photos taken with a single camera, as well as two multi-viewpoint cameras (Section 7). Finally, we describe the results of a user study with 145 subjects that compares the effectiveness of 3D vs. 2D pan & scan effects (Section 8).

2 Related work

The process of creating a 3D pan & scan effect is challenging and time consuming. There are a number of techniques that help in creating 3D fly-throughs from a single image, such as Tour Into the Picture [12] and the work of Oh et al. [24], though the task remains largely manual. Hoiem et al. [11] describe a completely automatic approach that hallucinates depths from a single image. While their results are impressive, substantially better results can be obtained with multiple photographs of a given scene.

To that end, image-based rendering (IBR) techniques use multiple captured images to support the rendering of novel viewpoints [14]. Our system builds a representation of a small portion of the 4D light field [19, 6] that can be used to render a spatially restricted range of virtual viewpoints, as well as sample a virtual aperture to simulate depth of field. Rendering novel viewpoints of a scene by re-sampling a set of captured images is a well-studied problem [1]. IBR techniques vary in how much they rely on constructing a geometric proxy to allow a ray from one image to be projected into the new view. Since we are concerned primarily with a small region of the light field, we are able to construct a proxy by determining the depths for each of the input images using multi-view stereo [34], similar to Heigl et al. [10] . This approach provides us the benefits of a much denser light field from only a small number of input images. Our technique merges a set of images with depth in a spirit similar to the Layered Depth Image (LDI) [29]. However, we compute depths for segments, and also perform the final merge at render time. Zitnick et al. [35] also use multi-view stereo and real-time rendering in their system for multi-viewpoint video, though they only allow novel view synthesis between pairs of input viewpoints, arranged on a line or arc. Most IBR systems are designed to operate across a much wider range of viewpoints than ours and typically use multiple capture devices and a more controlled environment [32, 18]. To date, the major application of capturing a small range of viewpoints, such as ours, has been re-focusing [21, 22].

A number of papers have used advanced graphics hardware to accelerate the rendering of imagery captured from a collection of viewpoints. The early work on light fields [19, 6] rendered new images by interpolating the colors seen along rays. The lightfield was first resampled from the input images. The GPU was used to quickly index into a lightfield data structure. In one of the early works leveraging per-pixel depth, Pulli et al. [26] created a textured triangle mesh from each depth image and rendered and blended with constant weights. They also introduced the notion of a soft-z buffer to deal with slight inaccuracies in depth estimation. We take a similar approach but are able to deal with much more complex geometries, use a per-pixel weighting, and have encoded the first soft-z into the GPU acceleration. Buehler et al. [1] rendered per-pixel weighted textured triangle meshes (one simple mesh per light field). We use a similar per-pixel weighting, but are also able to deal with much more complex and accurate geometries. We also use a "reverse soft-z" buffer to fill holes caused by disocclusions during rendering.

Subjects of interest	Camera moves	Image effects
0	Establishing dolly, Dolly-out	
1	Dolly in/out, Dolly zoom	Change *DOF*, Saturation/brightness
2	Dolly	Pull focus, Change *DOF*

Table 1: *A taxonomy of camera moves and image effects.* DOF *refers to depth of field.*

Automatic cinematography that follows common film idioms has been explored in the context of virtual environments, e.g., by He et al. [9]; we focus on the idioms used in 3D pan & scan effects.

3 3D pan & scan effects

Our first design goal is to automatically create 3D pan & scan effects that follow the conventions in documentary films. To that end, we examined 22 hours of documentary footage in order to extract the most common types of camera moves and image effects. We examined both films that employ 2D pan & scan effects (18.5 hours, from the Ken Burns films *The Civil War, Jazz,* and *Baseball*) and the more recent 3D pan & scan technique (3.5 hours, *The Kid Stays in the Picture,* and *Riding Giants*). These films contained 97 minutes of 2D effects and 16 minutes of 3D effects. Of these 113 minutes, only 9 exhibited non-linear camera paths; we thus ignore these in our taxonomy (though, as described in Section 4.4, curved paths can be created using our interactive authoring tool). Of the remaining 104 minutes, 102 are covered by the taxonomy in Table 1 and described in detail below (including 13 minutes that use a concatenation of two of the effects in our taxonomy).

We organize our taxonomy according to the number of "subjects of interest" in a scene: zero, one, or two. For each number there are several possible camera moves. There are also several possible image effects, such as changes in saturation or brightness of the subjects of interest or background, or changes in depth of field. These effects are typically used to bring visual attention to or from a subject of interest. The complete set of 3D pan & scan effects in our taxonomy includes every combination of camera move and image effect in Table 1 for a specific number of subjects of interest (e.g., no image effect is possible for zero subjects of interest). The most typical subject of interest used in these effects is a human face.

For scenes with no specific subject of interest, we observed two basic types of "establishing shots." These shots depict the entire scene without focusing attention on any specific part. In one type of establishing shot, the camera simply dollies across the scene in order to emphasize visual parallax. We will call this an *establishing dolly*. In the other type of establishing shot, the camera starts in close and dollies out to reveal the entire scene. We will call this an *establishing dolly-out*.

For scenes with a single subject of interest, two types of camera moves are commonly used. The first uses a depth dolly to slowly move the camera in toward the subject, or, alternatively to pull away from it. We will call this type of move a *dolly-in* or *dolly-out*. A variant of this move involves also reducing the depth of field while focusing on the subject to draw the viewer's attention. Another variant, which can either be combined with a changing depth of field or used on its own, is an image effect in which either the subject of interest is slowly saturated or brightened, or its complement (the background) desaturated or dimmed. The other type of camera move sometimes used with a single subject of interest is a kind of special effect known as a *dolly zoom*. The camera is dollied back at the same time as the lens is zoomed in to give an intriguing, and somewhat unsettling, visual appearance. This particular camera move was made famous by Alfred Hitchcock in the film, *Vertigo*, and is sometimes known as a "Hitchcock zoom" or "Vertigo effect." Like the other single-subject camera moves, this move works equally well in either direction.

Finally, for scenes with two primary subjects of interest, the camera typically dollies from one subject to the other. We call this move, simply, a *dolly*. There are two variations of this move, both involving depth of field, when the objects are at substantially different depths. In the first, a low depth-of-field is used, and the focus is pulled from one subject to the other as the camera is simultaneously dollied. In the other, the depth of field itself is changed, with the depth of field either increasing to encompass the entire scene by the time the camera is dollied from one subject to the other, or else decreasing to focus in on the second subject alone by the time the camera arrives there. In general, any of the camera moves for scenes with n subjects of interest can also be applied to scenes with more than n. Thus, for example, scenes with two or more subjects are also amenable to any of the camera moves for scenes with just one.

4 AUTHORING

In this section, we describe how to generate 3D pan & scan effects, initially focusing on automatically generated effects that follow our taxonomy, and concluding with an interactive key-framing system.

The input to this step is a light field with depth information. We assume that the input light field is sparse, and that novel views can be synthesized by projecting and blending textured depth maps. Due to the sparseness of the input, however, novel renderings will typically exhibit holes. While small holes can often be inpainted, large holes are best avoided. Computing a 3D pan & scan effect automatically from this input requires solving three problems. First, an effect appropriate for the imaged scene must be chosen from the taxonomy in Table 1. Second, a linear camera path must be computed that follows the intent of the effect and respects the limited sampling of the input. Third, any associated image effects must be applied.

4.1 Choosing the effect

Choosing an effect requires identifying the number of subjects of interest. In general, it is difficult, sometimes impossible, to guess what the user (or director) intends to be the subjects of interest in a scene. However, for our automatic system, a natural guess for a scene with people is to select their faces. We therefore run a face detector [33] on the centermost input view and count the number of faces. Then, one of the effects from the appropriate line in Table 1 is randomly chosen. The possible effects include image effects such as changing depth of field and focus pulls. Saturation and brightness changes, however, are left to the interactive authoring system, as they are less likely to be appropriate for an arbitrary scene.

4.2 Choosing a camera path

Each of the camera moves used in 3D pan & scan effects described in section 3 can be achieved by having the virtual camera follow a suitable path through camera parameter space. This parameter space includes the 3D camera location, the direction of the camera optical axis, and focal length. All of these parameters can vary over time. If we assume that all parameters are linearly interpolated between the two endpoints, the problem reduces to choosing the parameter values for the endpoints. The result is 6 degrees of freedom per endpoint — 3 for camera position, 2 for the optical axis (we ignore camera roll, uncommon in pan & scan effects), and 1 for focal length — and thus 12 degrees of freedom overall (two endpoints). A candidate for these 12 parameters can be evaluated in three ways.
1. The camera path should follow the particular 3D pan & scan convention.
2. The camera path should respect the limitations of the input. That is, viewpoints that require significant numbers of rays not sampled in the input should be avoided (modulo the ability to successfully fill small holes).
3. The camera path should be chosen to clearly exhibit parallax in the scene (as we show in our user study in Section 8, users prefer effects that are clearly 3D). Unfortunately, finding a global solution that best meets all 3 goals across 12 degrees of freedom is computationally intractable. The space is not necessarily differentiable and thus unlikely to yield readily to continuous optimization, and a complete sampling strategy would be costly, as validating each path during optimization would amount to rendering all the viewpoints along it.

We therefore make several simplifying assumptions. First, we assume that the second and third goals above can be evaluated by only examining the renderings of the two endpoints of the camera path. This simplification assumes that the measures for achieving those goals are generally greater at the endpoints than at points between them; e.g., a viewpoint along the line will not have more holes than the two endpoints, or at least not substantially more. While this assumption is not strictly true, in our experience, samplings of the space of viewpoints suggest that it often is. Second, we assume that the camera focal length and optical axis are entirely defined by the specific pan & scan effect, the camera location, and the linear interpolation parameter. For example, a dolly effect starts by pointing at the first subject of interest, ends by pointing at the second subject of interest, and interpolates linearly along the way. The focal length is set to keep the subjects of interest a certain size. As a result of these assumptions, the problem of choosing a camera path reduces to finding two 3D camera locations, such that the renderings of both viewpoints contain few holes and exhibit significant parallax relative to each other.

4.2.1 Valid viewpoint sampling

We first discuss how to identify viewpoints from which we can successfully render the input photographs processed by our system (measure #2). The set of all viewpoints can be described by a hypervolume parameterized by the camera parameters described above. We constrain this hypervolume to the finite region of valid viewpoints, where a valid viewpoint is defined as a viewpoint from which the rendered scene is complete or contains holes that are likely to be inpainted easily. We take advantage of the interactive renderer described in Section 6 to quickly determine valid viewpoints. Given the scene rendered from viewpoint V, we evaluate the success of the rendering using the following metric H:

$$H(V) = \frac{\sum_{p \in V} [d(p)]^k}{w \times h}$$

where w and h are the width and height of the rendering, $d(p)$ is the minimum distance from pixel p to a valid pixel, i.e., the boundary of the hole ($d(p)$ is set to 0 if p is not inside a hole), and k is a constant. Larger values of k penalize larger holes (which are harder to inpaint) over many smaller holes; we found $k = 3$ to be a be good value. We use a distance transform [4] to compute $d(p)$ quickly. We consider a viewpoint as valid if $H(V) < 2$.

The next step is to explore the hypervolume and define the 3D region of viewpoints that satisfy this viewpoint validity constraint. We assume that the coordinate system is aligned so that the input images mostly samples rays from viewpoints spread roughly across a plane (call it the x-y plane) looking in a direction (call it the z-axis) roughly perpendicular to that plane. Section 6 and Figure 3a describe this plane and associated view mesh in greater detail. We also assume that the centermost viewpoint is roughly at the origin of the coordinate system. For some effects, such as dolly-out, the camera motion is constrained to travel down the z-axis of this coordinate system. Other effects have more freedom; for these, we search for two regions of valid viewpoints, one for the starting and one for the ending camera viewpoints (since these will have different constraints on pointing direction and focal length).

To explore the range of valid viewpoints we uniformly sample along x and y at $z = 0$, and then search along z until $H(V)$ exceeds the threshold. The uniform sampling along x and y is done at a resolution of a 12×12 grid across a region that is 2 times the size of the bounding box of the viewpoints in the input light field (we have explored wider and denser samplings, and found these settings to be a good trade-off between efficiency and quality). We search in each direction along z until $H(V)$ exceeds the threshold. We search using an adaptive step size, increasing or decreasing the step size by a factor of two depending on whether the $H(V)$ threshold is met or exceeded, respectively. Figure 1 demonstrates a real example of such a grid (with a denser 100×100 sampling for visualization purposes), and the results of searching forwards in z.

4.2.2 Maximizing parallax

Next, we need an approach to measuring the parallax induced between two viewpoints. There are a number of possibilities for measuring parallax. We found that the most perceptually noticeable areas of parallax are those that are visible in one viewpoint and occluded in the other. We therefore project the starting viewpoint into the ending viewpoint and vice versa, and sum up the area of holes; this sum is our measure of parallax.

We assume that the starting and ending viewpoints will both be extremal in z; we thus have 144 candidates for both the starting and ending viewpoints (from sampling the 12×12 grid twice). We choose the highest scoring pair according to our measure of parallax. Since this measure requires projecting the starting viewpoint into the ending viewpoint and vice-versa, choosing the optimal pair would require $2 \times 144 \times 144$ projections, which is time-consuming. However, there is a strong correlation between the length of the camera path and the amount of parallax. We selected a dozen examples and performed the full set of projections. We found that the best pair of viewpoints by our parallax metric was always among the top 12 pairs when sorted by the length of the camera path. We therefore increase the speed by only performing the projections on the 12 longest camera paths, and choosing the one with the most parallax.

4.2.3 Constraints on the path

Each camera move in our taxonomy imposes specific constraints on the camera focal length, optical axis, and in some cases, camera

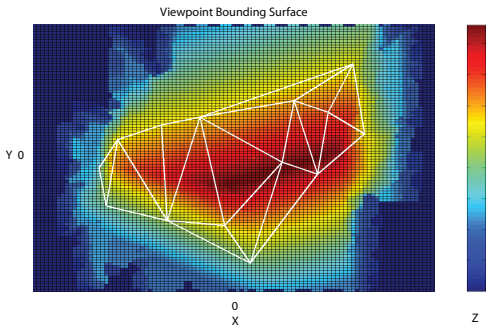

Figure 1: *A 100×100 grid of valid viewpoints color coded by z-value. A mesh connecting the original viewpoints in the input light field is shown in white. Note viewpoints in the central area are closer to the original sampled viewpoints; thus they can move much closer to the scene (larger z) than peripheral viewpoints.*

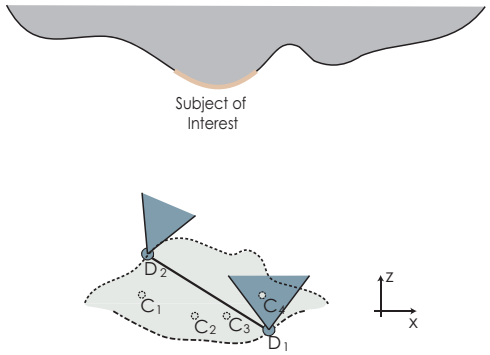

Figure 2: *For one subject of interest, our algorithm locates the linear camera path (D_1D_2) for dolly-in/out within the valid hypervolume. D_1 and D_2 are on the front and back edges of the valid hypervolume with respect to the z axis. Here C_x denotes the position of a viewpoint sampled in the input. Note how D_2 adjusts its orientation and focal length to keep the subject of interest centered.*

motion. These constraints are designed to mimic the effect's typical appearance as we observed them in documentary films. The constraints make use of information that we assume is contained in the input light field, such as the 3D centroid of the scene (straightforward to compute given that the input light field contains depth information), and the average focal length f of the capture device (in our case a standard camera). We now address each camera move, beginning with the case where there is no specific subject of interest.

Establishing dolly. The camera always points at the scene centroid, and the focal length is set to f for both ends of the camera path.

Establishing dolly-out. The camera moves along the z axis, always pointing at the scene centroid. The starting focal length is set to $1.5f$, and the ending focal length to f. In this case, no sampling grid across x, y is needed, (though search in the $+z$ direction and another in the $-z$ direction must be performed).

Next we consider the cases containing one or two subjects of interest. To test for the number of subjects of interest, we run the face detector, truncate to the two largest faces if more than two are found, and use the detected rectangles to construct a geometric proxy for the faces. We compute the median depth within each rectangle, and construct a 3D quadrilateral at that depth. Here we discuss 3D pan & scan effects for a single region of interest.

Dolly-in/out. In this case (Figure 2), the starting camera points at the lower half of the rectangle containing the subject of interest (so that the face is slightly above center), and the ending camera points at the scene centroid (or vice-versa). The focal length starts at $1.5f$ so that the face is zoomed-in, and ends with f (or vice-versa).

Dolly zoom. Here the algorithm follows the same procedure as when executing an "establishing dolly-out" with one exception: the focal length is adjusted during the animation to force the region of interest to have the same size as seen in the starting viewpoint.

Finally, we consider two regions of interest.

Dolly. The camera starts by pointing at the first subject of interest, and ends by pointing at the second. The focal length starts at $1.5f$, and ends at the same focal length times the ratio of the size of the subjects of interest (so that the final subject of interest ends at the same size as the first).

4.3 Image effects

For one or more subjects of interest our solution may choose to add depth-of-field and/or focus pull effects. Depth-of-field adds another degree of freedom per camera endpoint, namely the aperture diameter. After the two camera endpoints are chosen, a maximum aperture diameter must be chosen so that it does not change the validity of a viewpoint; we therefore search for this maximum. We assume that an aperture diameter can be evaluated by the maximum $H(V)$ of the four corners of the bounding box of the aperture. Starting with an initial maximum aperture diameter, we search adaptively for bigger apertures until one of the four corner viewpoints becomes invalid.

To add a changing depth of field to a dolly or dolly in/out, the aperture is linearly interpolated from a pinhole to the maximum aperture. For a focus pull, the aperture is kept at the maximum, and the in-focus depth is simply transitioned from the depth of one subject to the other.

4.4 Interactive camera control

We also allow a user to design camera paths that are outside of the taxonomy giving them the freedom to design novel camera paths, or to chain together and modify automatically generated paths. The user is free to navigate, using the interactive renderer, to desired viewpoints and adjust camera settings, as well as color effects (like saturation) and enter them as keyframes along a cubic spline path. Alternatively, the user can author paths by simply drawing one or more regions of interest, and choosing a camera move from the taxonomy; the system will then compute a path using the method for automatically detected regions of interest. Finally, the user can add depth-dependent brightness and saturation effects tied to a manually specified region of interest.

4.5 View morphing

For the special case where only two views are provided as input to the system, the method of Anonymous [34] computes stereo correspondences using a soft epipolar constraint. The soft constraint allows for some scene motion. The automatic system simply interpolates linearly between correspondences, i.e., morphs between the views, regardless of scene content.

5 Constructing Light Fields with Depth

The input light fields used to create our results are constructed using existing computer vision techniques from a few photographs of a scene taken from slightly different viewpoints, either by multiple shots from a standard camera, or with a single shot from a multi-viewpoint camera. First, a structure-from-motion system [31] is used to recover the relative location and orientation of each photograph, as well as the camera focal lengths.

Next, we compute a depth map for each viewpoint using a multi-view stereo algorithm [34]. Like some other top-performing stereo algorithms [16, 27], this method performs an over-segmentation of the input photographs and then determines a depth per segment, rather than a depth per pixel. In our representation, within a single view, the constant-z segments have spatially varying colors and can overlap each other, at most two segments covering each pixel, with blending weights that sum to one at each pixel. We can express the color and weight variations within a segment using RGBA textures. By rendering these segmentation-based depth maps into novel viewpoints, we can effectively "hallucinate" the light field in the neighborhood of the input views.

6 Rendering Algorithms

We have developed two rendering algorithms to display novel views from the textured segments with depth. The first rendering algorithm is implemented as a real-time renderer leveraging the GPU. The second algorithm is implemented as an off-line renderer and produces higher quality results; it is used to render the final sequences. Both renderers take the same basic approach; we therefore first describe the general rendering algorithm, and then we describe the specifics of the interactive renderer implementation including GPU acceleration, and finally we describe the differences in the off-line rendering algorithm.

The first step for either rendering algorithm is to construct a *view mesh*, a triangle mesh with vertices at the centers of projection (COPs) of the input views. To construct the mesh, we fit a plane to the COPs, project them to that plane, and perform a Delaunay triangulation [30]; the connectivity of this triangulation is used, but the original COP locations are retained. We extend the view mesh by duplicating the vertices on the mesh boundary (Figure 3a). These duplicate vertices are positioned radially outward from the mesh center at a distance four times the distance from the center to the vertices on the boundary.

To render a single ray from a novel viewpoint, we intersect the ray with the view mesh. The intersected triangle indicates which three views will contribute to the ray, and we use the barycentric coordinates of the intersection as the *viewing weights* associated with the corresponding views. The ray is then intersected against the segment geometry in each of the three views. Because each view has independent reconstructions of scene depths, the same surface point may be reconstructed with somewhat different depths. We combine the depths and colors along the ray with a *soft z-buffer* to resolve depth inconsistencies [26]. In the end, each color is weighted by a product of its segment weight, viewing weight, and soft-z weight; the weighted colors are summed and divided by the sum of the weights.

For views that are too far from the input views, some rays will intersect no segments; this is most prevalent around depth discontinuities. The resulting holes in the rendering are filled by inpainting.

6.1 Interactive renderer

To render the scene from a novel view at interactive frame rates (at least 30 fps), we developed a GPU-based algorithm. We render the scene in four steps. First, we choose which views should contribute to a pixel value and calculate the per-pixel viewing weight. Second,

we render all of the segments to three offscreen buffers. Third, we employ a soft z-buffer to resolve depth inconsistencies between the three offscreen buffers and combine their color values. Finally, we fill holes using a *reverse soft z-buffer* and local area sampling.

Rendering the extended view mesh To choose which segments contribute to the pixel value and to calculate the viewing weights, we render the extended view mesh from the novel viewpoint to an offscreen buffer. We encode both viewpoint IDs and the barycentric weights into the vertex attributes (e.g. color or normal data), so that the rendering can be used determine which viewpoints contribute to a pixel and the corresponding viewing weights.

When rendering the extended view mesh, there are two special cases that should be highlighted. First, if the novel view lies in front of the view mesh, the projection step requires a backwards projection. Second, as the novel view moves close to view mesh, we must take care to avoid a singularity. In this case we use the blending weights and viewpoint IDs of the nearest point on the mesh.

Rendering segments Next the image segments will be rendered and combined. First, we note that each final pixel color will ultimately have contributions from three viewpoints. Thus, we create three off-screen, RGBA *rendering buffers*. We associate with each pixel in each rendering buffer a single viewing ID that indicates the viewpoint that contributes to the pixel (exactly one viewpoint will contribute to any given pixel in a rendering buffer).

Each segment is rendered as a texture-mapped rectangle. However, it would be quite inefficient to store each segment in a separate small texture.[2] Recall that each input image is decomposed into segments that overlap, with at most two segments covering any pixel. We store two segment images per input view; each pixel in each segment image has an RGBA color and weight and asociated segment ID. If an input pixel is covered by only one segment, then one of the segment images has zero weight (and arbitrary color and segment ID) at the corresponding pixel location. In practice, the RGBA components are stored in two textures, and the segment IDs are stored in one 32 bit texture, with 16 bits per ID.

The 3D rectangle and texture coordinates for a segment are determined by the bounding box of the segment. Before rendering the bounding rectangle, we encode segment ID and viewpoint ID as vertex attributes. The shader uses these attributes in conjunction with the segment image textures to render the segment color on the fly. In particular, when rendering a given segment from a given viewpoint, the pixel shader only selects for the segment image color with the matching segment ID, and composites it into the rendering buffer with the matching viewpoint ID (if any).

Before rendering any segments, the segments for each viewpoint are sorted in front-to-back order. The three rendering buffers are initialized to black background and zero alpha. Then, all segments in each viewpoint are rendered in front-to-back order, weighted by their alphas.

Soft z-buffer We employ a *soft z-buffer* to combine the three rendering buffers. We compute a soft-z weight w_z at each pixel by comparing each pixel's z-value with the z-value of the pixel closest to the origin of the novel view. This distance Δz, where $\Delta z = z - z_{\text{closest}}$, is used to compute w_z in the following equation:

$$w_z(\Delta z) = \begin{cases} 1 & \text{if } \Delta z \le \gamma \\ \frac{1}{2}(1+\cos\left(\frac{\pi(\Delta z-\gamma)}{\rho-2\gamma}\right)) & \text{if } \gamma < \Delta z \le \rho-\gamma \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

[2]We tried this storage mechanism originally, and the performance was roughly ten times slower than what we describe in this section.

where ρ is the depth range (max$-$min) of the entire scene, and γ is set to $\rho/10$.

The set of w_z's are normalized to sum to one. The depth z is then given the sum of the z-values weighted by the w_z's. Each viewing weight stored in the view mesh texture is multiplied by its corresponding w_z. These new blending weights are normalized. The final pixel value is computed by scaling each pixel by the normalized blending weight, and combined based on their alpha values.

Hole filling Holes occur when, due to parallax, a nearby segment separates from a more distant segment. We fill small holes of less than 6 pixels in diameter during the final *soft z-buffer* pass. Since holes occur due to disocclusion, given two neighboring pixels, we prefer to use the more distant one to fill the gap. To do so, we compute a weighted sum of pixel colors and z-values of the pixels in a 7×7 neighborhood. The weights are taken from those used in the soft z-buffer calculation described above except "in reverse." In particular, more distant z-values are given higher weights by setting the z-values to $1 - z$.

Color effects and depth of field Image color effects such as selective brightening and/or desaturation to 3D are constrained to a 3D bounding region. These effects are potentially any image + depth operations, including saturation, color shifting, brightening, or relighting. We model the effect after the layer paradigm commonly used in image and video processing tools such as Adobe Photoshop or After Effects. Each effect is considered to be an independent layer composited over the rendered image. If the user applies more than one effect, its layer is composited in a user specified order.

The effect parameters vary at runtime and include the bounding 3D region, two texture maps consisting of the rendered scene and the z-buffer, and a boundary compositing parameter. The boundary parameter allows seamless blending in scenes with a noisy depth map. This parameter controls effect layer compositing by specifying a Gaussian falloff from the user-specified 3D bounding region. Each effect may have additional parameters to control its application. Finally, each user editable effect parameter can be specified with keyframes so that the effect can animated along with the camera path.

Efficient, approximate depth-of-field rendering is accomplished in this framework using a variation on existing methods [3, 15, 7]. For each pixel, we calculate a circle of confusion based on a user defined aperture size, and blur the result of our rendered scene accordingly. The blurring operation is performed efficiently by loading the scene into a MIPMAP and indexing into it based on the blur kernel radius. To improve visual quality, we index into a higher resolution level of the MIPMAP than strictly needed, and then filter with a Gaussian filter of suitable size to achieve the correct amount of blur. Note that when focusing on the background in a scene, this approach will not result in blurred foreground pixels that partially cover background pixels as they should, i.e., the blurry foreground will have a sharp silhouette.

To avoid such sharp silhouettes, when processing a pixel at or behind the focus plane, the pixel shader blends the pixel with a blurred version of the image at that pixel. The blur kernel size is based on the average z-value of nearby foreground pixels. The blending weight given to this blurred version of the image is given by the fraction of the neighboring pixels determined to be foreground. The size of the neighborhood is determined by the circle of confusion computed from the user specified aperture and focal depth.

6.2 Off-line rendering

The higher quality off-line rendering algorithm differs from the interactive renderer in three main ways. First, we extend the *soft z-*

buffer described above to increase the accuracy of our pixel value estimate. Second, the renderer uses a texture synthesis approach to fill any holes and cracks that might appear in a novel view due to sparse data generated from the input photographs. Finally, depth of field effects are rendered with increased quality by simulating a camera aperture.

Soft z-buffer The soft z-buffer calculation is very similar to the process described in the real-time renderer. However, rather than using a traditional hard z-buffering within each viewpoint followed by a soft z-buffer across viewpoints, all segments from all contributing viewpoints are combined in a uniform manner.

Hole filling To fill holes the offline renderer uses an approach similar to the in-painting algorithm of Criminisi et al. [2] with two modifications. First, to accelerate computation, we search for matching (5×5) neighborhoods within a restricted window (100×100) around the target pixel. The second, more significant, modification is based on the observation that nearly all large holes occur along depth discontinuities, because some region of background geometry was always occluded in the input photographs. In this case, the hole should be filled from background (far) regions rather than foreground (near) regions. We thus separate the depths of the pixels along the boundary into two clusters, and use these two clusters to classify pixels, as needed, as foreground or background. We fill the hole with Criminisi's propagation order, using modified neighborhoods and neighborhood distance metrics. When copying in pixel color, we also inpaint its z by weighted blending from known neighbouring pixels, again favoring the back layer. The inpainted z assists in region selection for color manipulation effects. Note that Moreno-Noguer et al. [21] also explored depth-sensitive inpainting, though their application has lower quality requirements since they use the inpainted regions for rendering defocused regions rather than novel viewpoints.

Depth of field Our rendering algorithm now provides a way to reconstruct any view within the viewing volume. In addition to changing viewpoint, we can synthetically focus the image to simulate camera depth of field. To do so, we apply an approach similar to what has been done in *synthetic-aperture photography* [19, 13]. We jitter viewpoints around the center of a synthetic aperture and reconstruct an image for each viewpoint. We then project all the images onto a given in-focus plane and average the result.

7 Results

We have tested our overall approach (including the construction of the light field from a few photographs) on 208 datasets capturing a variety of scenes. About half of the datasets (103 out of 208) produced successful results that were comparable to the results shown in this paper. The failures were largely due to subject motion and errors in structure-from-motion and multi-view stereo.

A subset (20) of the successful results are included in the accompanying video. Only 2 examples were interactively authored. The rest were generated entirely automatically through our pipeline. The best way to experience the cinematic effects produced by our system is in animated form, as shown in the supplemental video. The number of input images ranges from 8 to 15 and are typically captured with just 3-4 inches of space between the viewpoints. Most datasets are captured at resolution 1200×800. A 3GHz PC with 4GB memory and an NVIDIA 8800 class GPU interactively renders scenes at 30-45 frames-per-second. Automatically generating a 3D pan & scan effect automatically takes about 3 hours, including 10-15 minutes for structure-from-motion, 100-120 minutes for multi-view stereo, 2-5 minutes for computing the effect, and 25-35 minutes for the off-line rendering.

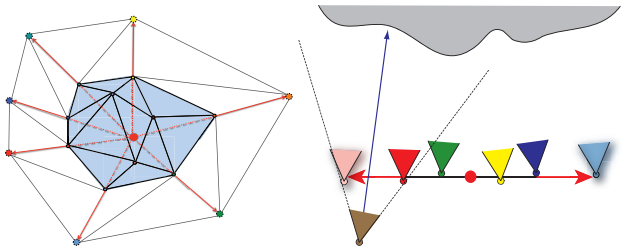

Figure 3: (a) *The view mesh is extended by creating copies of boundary vertices and placing them radially outward (red arrow) from the centroid (red dot) of all of the mesh vertices. The dotted blue arrow shows a ray projected through the image plane. The* blending weights *at this pixel correspond to the barycentric coordinates of the intersected triangle of viewpoints in the view mesh.*

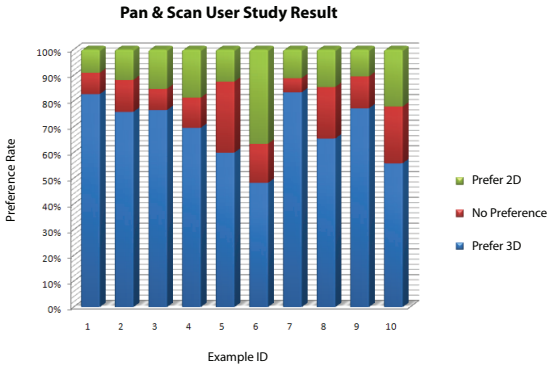

Figure 4: *The preference rate of the 10 examples in the user study. Note that fewer than* 65% *of the subjects preferred the 3D version of examples 5, 8 and 10; these examples contained the most subtle parallax or motion, and thus support the notion that maximizing parallax is important.*

While some users may be willing to capture multiple viewpoints, a single-shot solution is certainly more appealing and would address the problem of subject motion between viewpoints. To that end, we include two examples from multi-viewpoint cameras: an integral-lens camera [5], and the $50 Pop9 film camera.[3] The results are not perfect, notably for the prototype integral-lens camera, due to chromatic aberrations in the prisms used to capture 20 images at once. However, these results demonstrate that the small baselines of these cameras can still yield subtle but noticeable parallax. In addition, we include two results using two input views with small scene motion (the guitarist and the smiling girl), for which our automatic system generates morphs from one view to the other.

Overall, the sequences are smooth and convey mild to strong motion parallax. However, occasional artifacts can be seen, often near depth boundaries. These artifacts come from depth misestimates; as multi-view stereo algorithms improve these types of errors should be reduced.

8 User Study

In order to assess the perceptual quality of the 3D pan & scan effects, we conducted a user study study with 145 subjects, the majority of whom were in the IT industry but not in the graphics industry. Each subject was asked to watch 10 examples, with each example shown in two versions: a 2D pan & scan, and a 3D pan & scan. All

[3] http://shop.lomography.com/pop9/

effects were rendered at 480×320 pixels at 30 frames per second and were 3-5 seconds long. For each example, subjects had to indicate whether they noticed any difference between the two versions after watching them as many times as they liked. They also had to select which version they preferred, and optionally provide reasons for their choices.

The results show that there was a significant perceptual difference between 3D pan & scan and 2D pan & scan effects. Overall, the two versions of each example were judged to be different 94% of the time, and at least 80% of subjects noticed at least some difference between the two versions on every example.

The results also show that the majority of participants prefer the 3D versions to the 2D ones. Summing over all examples, 70% of subjects, on average, preferred 3D pan & scan, 16% preferred 2D pan & scan, and 14% had no preference. As shown in Figure 4, the preference rate exhibits correlations with the amount of parallax contained in each example. Note also that roughly half the people preferred the 2D to the 3D "dolly-zoom" example (#6 in Figure 4); in written comments they noted that the 3D effect was too dramatic and uncomfortable, which is the intended purpose of the effect. The users' written comments clearly show that apparent parallax was the dominant reason (69% of all reasons collected) behind the preference for the 3D results. Taken together, the results of the user study clearly indicate that the cinematic effects we created offer a more compelling depiction than a simple 2D pan & scan.

9 Conclusion

Recent advances in computational photography have dramatically increased the amount of information that we can capture from a scene. Until now, techniques that capture depth along with an image have been used primarily for digital re-focusing, on the assumption that small parallax changes are uninteresting. On the contrary, we believe that subtle parallax can lead to a richer, more informative visual experience of a scene that feels fundamentally different than a still photograph or 2D pan & scan. As multi-view camera designs and computer vision techniques continue to improve, we see an opportunity for parallax photography to become a widely used approach to capturing the moments of our lives.

References

[1] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured lumigraph rendering. In *Proc. of SIGGRAPH 2001*, pages 425–432, August 2001.

[2] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, Jun. 2003.

[3] J. Demers. Depth of field a survey of techniques. In R. Fernando, editor, *GPU Gems 1*, chapter 23, pages 375–390. Mar. 2004.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell University, 2004.

[5] T. Georgiev, C. Zheng, S. Nayar, D. Salesin, B. Curless, and C. Intwala. Spatio-angular resolution trade-offs in integral photography. *Proc. of Eurographics Symposium on Rendering*, 2006.

[6] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. B. Cohen. The lumigraph. *ACM Trans. Graph.*, pages 43–54, 1996.

[7] E. Hammon. Practical post-process depth of field. In H. Nguyen, editor, *GPU Gems 3*, chapter 28. Addison Wesley, 2007.

[8] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[9] L.-W. He, M. F. Cohen, and D. H. Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proc. of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 217–224, Aug. 1996.

[10] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. J. V. Gool. Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *DAGM-Symposium*, pages 94–101, 1999.

[11] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 24(3):577–584, August 2005.

[12] Y. Horry, K.-I. Anjyo, and K. Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Proc. of SIGGRAPH 97*, pages 225–232, August 1997.

[13] A. Isaksen, L. McMillan, and S. J. Gortler. Dynamically reparameterized light fields. *ACM Trans. Graph.*, pages 297–306, 2000.

[14] S. B. Kang and H.-Y. Shum. A review of image-based rendering techniques. In *IEEE/SPIE Visual Communications and Image Processing 2000*, pages 2–13, 2002.

[15] M. Kass, A. Lefohn, and J. D. Owens. Interactive depth of field using simulated diffusion. Technical Report 06-01, Pixar Animation Studios, Jan. 2006.

[16] A. Klaus, M. Sormann, and K. F. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *International Conference on Pattern Recognition (ICPR)*, pages 15–18, 2006.

[17] A. Levin, R. Fergus, F. Durand, and W. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graph.*, 26(3):70:1–70:9, July 2007.

[18] M. Levoy. Light fields and computational imaging. *IEEE Computer*, 39(8):46–55, 2006.

[19] M. Levoy and P. Hanrahan. Light field rendering. *ACM Trans. Graph.*, pages 31–42, 1996.

[20] C.-K. Liang, T.-H. Lin, B.-Y. Wong, C. Liu, and H. Chen. Programmable aperture photography: Multiplexed light field acquisition. *ACM Trans. Graph.*, 27(3), 2008.

[21] F. Moreno-Noguer, P. N. Belhumeur, and S. K. Nayar. Active refocusing of images and videos. *ACM Trans. Graph.*, 26:671–679, 2007.

[22] R. Ng. Fourier slice photography. *ACM Trans. Graph.*, 24(3):735–744, Aug. 2005.

[23] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. Technical Report CSTR 2005-02, Stanford University, 2005.

[24] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *Proc. of SIGGRAPH 2001*, pages 433–442, August 2001.

[25] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, Sept. 2004.

[26] K. Pulli, M. F. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle. View-based rendering: Visualizing real objects from scanned range and color data. In *Eurographics Rendering Workshop 1997*, pages 23–34, June 1997.

[27] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[28] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 519–528, June 2006.

[29] J. Shade, S. J. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *Proc. of SIGGRAPH 98*, pages 231–242, July 1998.

[30] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

[31] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006.

[32] D. Taylor. Virtual camera movement: The way of the future? *American Cinematographer*, 77(9):93–100, Sept. 1996.

[33] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[34] K. C. Zheng, A. Colburn, A. Agarwala, M. Agrawala, B. Curless, D. Salesin, and M. Cohen. A consistent segmentation approach to image-based rendering. Technical Report CSE-09-03-02, University of Washington, 2009.

[35] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, Aug. 2004.