PlenoPatch: Patch-based Plenoptic Image Manipulation

Fang-Lue Zhang, *Member, IEEE,* Jue Wang, *Senior Member, IEEE,*
Eli Shechtman, *Member, IEEE,* Zi-Ye Zhou, Jia-Xin Shi, and Shi-Min Hu, *Member, IEEE*

Abstract—Patch-based image synthesis methods have been successfully applied for various editing tasks on still images, videos and stereo pairs. In this work we extend patch-based synthesis to plenoptic images captured by consumer-level lenselet-based devices for interactive, efficient light field editing. In our method the light field is represented as a set of images captured from different viewpoints. We decompose the central view into different depth layers, and present it to the user for specifying the editing goals. Given an editing task, our method performs patch-based image synthesis on all affected layers of the central view, and then propagates the edits to all other views. Interaction is done through a conventional 2D image editing user interface that is familiar to novice users. Our method correctly handles object boundary occlusion with semi-transparency, thus can generate more realistic results than previous methods. We demonstrate compelling results on a wide range of applications such as hole-filling, object reshuffling and resizing, changing object depth, light field upscaling and parallax magnification.

Index Terms—Plenoptic image editing, light field, patch-based synthesis

1 Introduction

Light field (or plenoptic) cameras have significantly advanced and become wide spread in recent years, showing a high potential to revolutionize photography. Such cameras typically use microlens arrays to capture 4D light field information of the scene, resulting in a plenoptic image that encodes scene appearance from different viewpoints. Since it contains 3D scene information, a plenoptic image enables new post-processing possibilities that are impossible for conventional 2D images, such as changing the viewpoint and refocusing. As recent consumer-level plenoptic cameras such as Lytro [1] ,PiCam [48] and Raytrix [2] are gaining popularity, there is an increasing demand for advanced editing tools for manipulating their outputs.

Despite the rapid development in hardware design of plenoptic cameras, plenoptic image editing has been a less explored territory. Previous approaches have explored a limited range of editing operations [21] such as super-resolution [4], morphing [50], view c synthesis [27], refocusing [36] and retargeting [3]. In contrast, 2D image editing has been an extensively studied area and among numerous available techniques, patch-based synthesis methods have emerged as the state-of-the-art for many applications such as hole filling, retargeting and reshuffling [2]. In this paper, we extend patch-based methods for interactively manipulating consumer plenoptic images, to achieve new light field editing effects that were not explored with existing techniques, such as removing an object from the scene, locally editing and reshuffling objects, changing the depth of an object and increasing the depth range of the scene through parallax magnification. Our method also generates better light field upscaling results than existing methods.

Our system is designed for consumer light field data. We

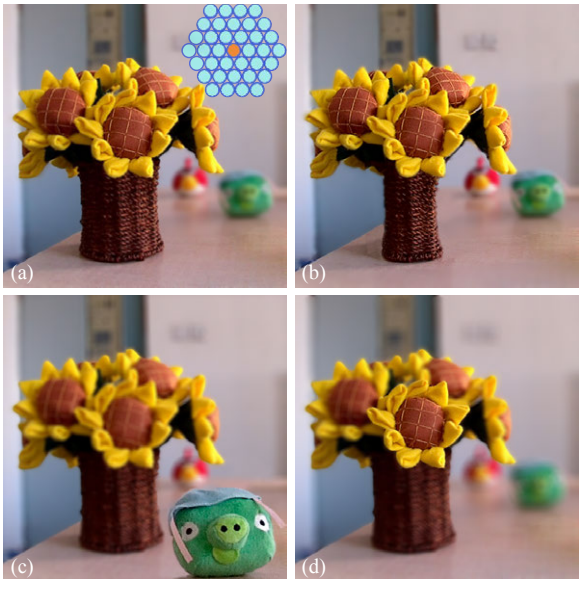


Fig. 1. Patch-based plenoptic image editing. (a) A refocused central view of the input light field. (b) Shrinking the flower bouquets. (c) Moving an object towards the camera. (d) Narrowing the depth-of-field by parallax magnification (Bokeh effect).

- *Fang-Lue Zhang, Zi-Ye Zhou, Jia-Xin Shi and Shi-Min Hu are with TNList, Tsinghua University, Beijing, China. E-mail: z.fanglue@gmail.com, jerry.zhou@gmail.com, ishijiaxin@qq.com and shimin@tsinghua.edu.cn. Shi-Min Hu is the corresponding author.*
- *Jue Wang and Eli Shechtman are with the Creative Technologies Lab of Adobe Research, Seattle, WA 98103 USA., E-mail: juewang@adobe.com and elishe@adobe.com*

1. https://www.lytro.com
2. http://www.raytrix.de

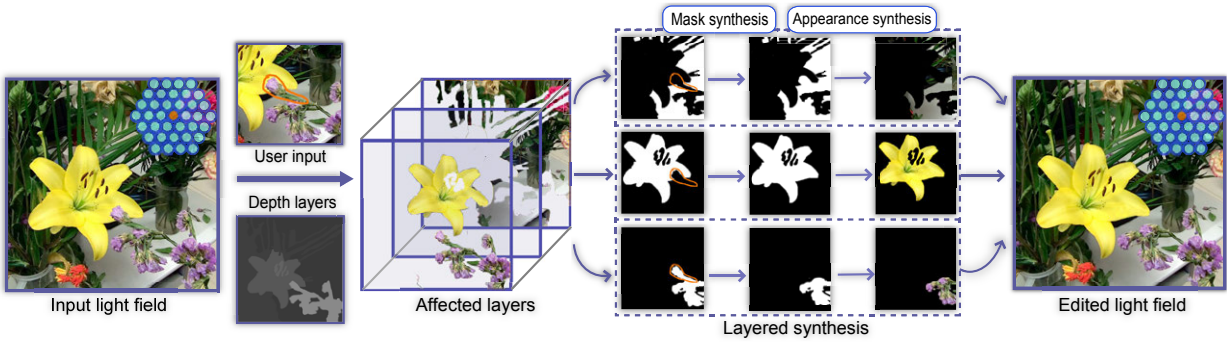


Fig. 2. The flowchart of the proposed interactive light field editing system. The light field is represented by a set of images captured at 37 viewpoints with a hexagonal layout, visualized in the top-right corner of the left image. Given an input light field, our system extracts a depth layered representation at the central view, where the user specifies the editing task. Patch-based synthesis is then applied to all layers that are affected by the task to create the edited version of the central view. Finally, the edits are propagated to all other views to generate the full modified light field.

therefore take advantage of the fact that the scene content of such data can be approximately treated as *overlapped layers*, a common approximation in consumer lenselet-based cameras. The view angle differences in the lenselet array of a consumer camera are small compared to large camera array prototypes, making the same objects have very similar appearance across different views. This is also the reason why the Lambertian assumption can be used to recover the light field from the raw data of lenselet-based cameras. In our work we thus represent the light field as a set of sub-images arranged by the coordinates of the lenses, and each image can be decomposed into a few depth layers.

As shown in Figure 2, given an input light field, we first recover the depth layers for the central view image (Section 4), which is then presented to the user for specifying an editing task using rough scribbles, providing a natural user interface that is similar to common interfaces in existing photo editing tools. Given the user-specified task, we identify all layers that are affected by it, and apply patch-based, depth-layer-aware image synthesis on all affected layers to meet the editing specifications (Section 5). Finally, edits made on the central view are propagated coherently to all other views to generate the final modified light field (Section 6).

Our main technical contributions include:

1) A novel framework that enables applying advanced patch-based image synthesis methods for plenoptic image editing, through a conventional 2D image editing user interface;
2) A layered synthesis method for depth-aware image editing (of the central view);
3) A method for propagating edits from the central view to other views while maintaining view point consistency;
4) A variety of new light field editing applications that are made possible by the proposed system, including hole filling, reshuffling and cloning, change of depth and parallax magnification.

2 Related Work

Light field capture and analysis. Earlier systems use large camera arrays for light field capturing [28], [54], while Ng *et al.* [36] developed the first hand-held light field camera. The light field can also be captured by microscopes [29]. With the recently developed compact light field cameras such as PiCam [48] and Lytro, plenoptic images have become accessible to consumers. To process the raw data captured by such devices, Dansereau *et al.* [7] proposed a parametric plenoptic camera model and a corresponding procedure for decoding, calibrating and rectifying plenoptic data. There have also been a series of works on how to display high quality light fields [52][32][25][20][14] and reconstruct them [26][44], and how to generate stereoscopic views [18] and stereo images [23] from them. These approaches are mainly concerned with capturing and displaying the light field rather than interactive light field editing, the main goal of this work.

Ng [35] and Egan *et al.* [11] analyzed the light field data in the 4D Fourier domain, which leads to more recent works on using the spectrum domain methods for analyzing the dimensionality gap and recovering the full light field from the focal stack based on the Lambertian assumption [27] [24]. However, Fourier methods do not support precise local object editing, which is the goal of our work. Recently, Goldluecke et al. [16] and Wanner et al. [51] propose methods to analyze the structure and perform segmentation in light fields, which do not focus on interactive editing either.

Plenoptic and Stereo image editing. There have been some previous approaches on interactive light field editing. Seitz and Kutukakos [40] and Horn and Chen [19] proposed interactive systems for directly editing the light rays while preserving the consistency of the scene when viewing the light field from different perspectives. Various interactive methods for morphing 3D objects represented by light fields [50] [6] [55] and stabilizing light field video [45] have also been presented. However, these methods do not support more advanced editing goals such as completion

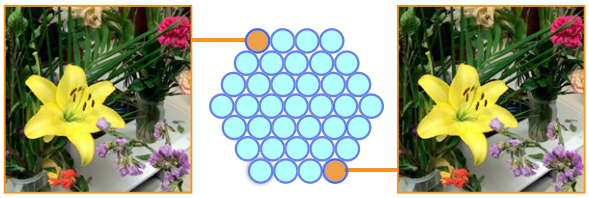


Fig. 3. The hexagonal layout of the camera array.

and reshuffling. Jarabo *et al.* [22] proposed a novel affinity function that can be used to propagate sparse user edits to the full light field, and presented a thorough study to evaluate workflows of related editing tasks [21]. Birklbauer and Bimber [3] presented a z-stack seam carving method for light field retargeting. One major limitation of the above work is that they adopt a holistic approach (e.g., using a global deconvolution) to recover the edited light field, which often leads to visual artifacts when rendering regions near depth edges. Our method carefully handles object boundaries to avoid producing such artifacts. These approaches are also aiming a specific editing task and are not general enough to be applicable for other tasks.

Our work is also related to stereo image editing. Various editing tools such as object cut-and-paste [31] [47] [33], resizing [30] and warping [37] have been developed for stereo image pairs. Patch-based methods have also been explored for hole filling in stereo pairs [34], but this method cannot be directly extended to plenoptic images. This is because it uses a cross-image patch searching and matching, while a plenoptic image contains so many views that would make the search space intractable. We tried a more elaborate extension of their method but found that it still does not provide enough consistency across views comparing to our method. A recent method by Boominathan *et al.* [5] applied patch-based approach for the upsampling (super-resolution) a plenoptic image, an applications we show as well. Similarly to [34], these patch-based methods do not handle properly depth boundaries, where there are pixels that are only visible in certain views. We compare to both methods in Section 7 and 8.

Patch-based image editing. Non-parametric patch-based synthesis methods have advanced substantially in recent years and achieved state-of-the-art performance in various image and video editing tasks. These include image-based rendering [12], hole-filling [53], texture synthesis [10], retargeting and reshuffling [9], super-resolution [15], morphing [43], image compositing and interpolation [8] and HDR reconstruction [41]. In particular, Barnes *et al.* [2] proposed a patch-based fast approximate nearest-neighbor search algorithm called PatchMatch, enabling image editing at interactive rates. The success of this approach has led to implementations in popular commercial tools such as Adobe Photoshop [1] and other image editing software. These methods however were not designed for plenoptic image editing.

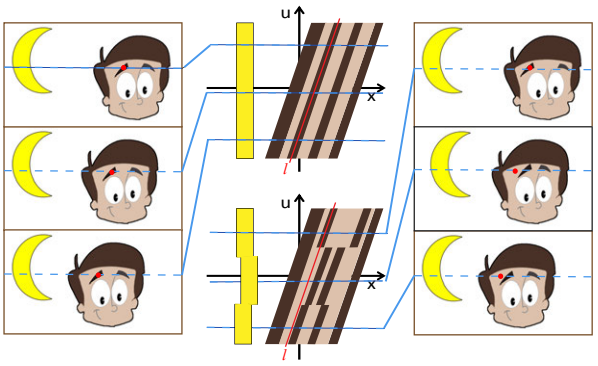


Fig. 4. 2D (x, u) slices for an original light field (top) and an edited version that is not coherent across views (bottom), and the corresponding images on the left and right. In the top slice, the highlighted scene point on the eyebrow is on a line that has the same color across different views. In the wrongly-modified light field, pixel colors along the same line contain sudden changes, corresponding to visual discontinuities and incorrect depth perception.

3 Overview

A continuous light field is represented discretely by a set of images $I_{u,v}(x, y)$: each captured by a single camera from a view (u, v), where the intensity (or color) value at pixel (x, y) in $I_{u,v}$ is $L(x, y, u, v)$. Each image $I_{u,v}(x, y)$ contains a set of pixels that are arranged in a rectangular 2D grid.

Several constraints need to be taken into consideration when editing a light field. First, we need to consider *fidelity*, meaning that an edited view image should have similar appearance with the corresponding input view. Secondly, we want to preserve *cross-view coherence* to ensure that the edits in neighboring views do not introduce visual discontinuity so the viewer perceives a realistic scene when switching between views. We propose a method to edit the light field while considering the fidelity and cross-view coherence, as shown in Figure 2. To describe fidelity mathematically, similar to previous patch-based image manipulation methods [2], for each view (u, v), we compute a fidelity measure as the sum of the differences between every patch in the edited image $I'_{u,v}$ and its most similar patch in $I_{u,v}$, where the patch differences D are measured using SSD (Sum of Squared Differences) in Lab color space as:

$$E_f(u, v) = D(A_t, A_s), \tag{1}$$

where $D(s, t)$ is the depth-aware patch difference function that will be defined in Section 4. A_t and A_s refer to the edited region in $I'_{u,v}$ and the source region in $I_{u,v}$, respectively. Detailed explanations will be given in Section 4.

Cross-view coherence means that a point on a real scene should have very similar appearance when observed from

different views. In the original light field, for every point (x, y, u, v) in the 4D space, the points that map to the same scene point should be on one plane that passes though it, on which the pixel values are near constant. Figure 4 illustrates the 2D projection of 4D light field on the (x, u) plane, where the lines are the projection of those planes. The slope of each line is determined by the distance between the scene point and the camera. If the editing operations violate cross-view coherence, these lines might break, resulting in sudden visual jumps when switching views. This is shown in the bottom of Figure 4. Formally, we define the *cross-view coherence* for each point (u, v, x, y) as:

$$E_c(u,v) = \sum_{p \in I'_{u,v}} \min_d \sum_{m,n} ||I'_{u,v}(p) - I'_{m,n}(p')||,$$
$$\text{with } p = (x, y),\ p' = (x + d(m-u), y + d(n-v)), \tag{2}$$

where d is the disparity value, p is a point in the current view (u, v) and p' is a corresponding point according to the disparity, in a different view (m, n). We first find the disparity of p, i.e. the slope of its coherent line along which the appearance variation is minimal, and then evaluate the view consistency along this line. To minimize the impact of color noise, we use a local $5 * 5$ patch instead of a single pixel to measure the view consistency by calculating the L_2-norm between the patches. Finally, we sum over all patches in $I'_{u,v}$ to derive the coherence measurement: a large value indicates a violation of the cross-view coherence constraint. In this cost function, the variables to be optimized are the edited pixel value $I'_{u,v}(p)$.

Our method firstly minimizes E_f for the central view in the edited light field using a modified layer-based PatchMatch method (see Section 5). Then the edits are propagated to other views based on the layered scene approximation, resulting in low E_c for the modified light field (see Section 6.1). Finally, we use a sub-pixel refinement method to further minimize E_c across all views (see Section 6.2).

4 Layered Scene Representation

4.1 Justification

Our light field editing framework is based on the assumption that the captured light field data can be well approximated by a fixed number of scene layers at different depths. While this is not generally true for any light field data, we have found that it holds well for images captured by consumer-level light field cameras such as Lytro. To quantitatively evaluate how well the layered representation can approximate the original data captured by such camera, we conduct the following experiment.

We extract different numbers of depth layers from the original light field data using the optimization approach proposed in Section 4.2. The number of scene layers ranges from 1 to 128. We then use the extracted depth layers to reconstruct the light field using the view synthesis approach proposed in Section 7.3, and constrain the target views to be exactly the same as the input views. We compute the average per-pixel reconstruction error as the means to evaluate the accuracy of the layered representation. The results for some examples and the average errors for all examples used in this paper are plotted as curves in Figure 5.

The results show that when the number of depth layers are small (less than 8), the reconstruction error quickly drops as the number of scene layers increases, indicating that more accurate representation is achieved. However, the reconstruction error becomes stable after reaching 32 depth layers, and adding depth layers beyond that does not help. This experiment suggests that the light field data produced by the Lytro camera may have already been compressed and is possibly represented internally using a finite set of discrete layers. This is a reasonable choice given that the camera has narrow parallax.

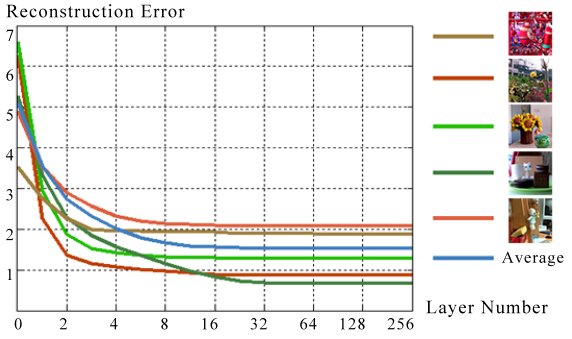


Fig. 5. The image reconstruction errors using different number of scene layers to represent the original light field data captured by a Lytro camera.

4.2 Layer Extraction

The layered representation consists of a series of soft layer masks M^l defined in the central view image $I(0, 0)$. Each layer is associated with a disparity value d_l, which can be used for transferring the mask of the current layer to other views.

We first use the cost-volume filtering method [38] to generate an initial layered depth map for $I_{0,0}$. Suppose we have L depth layers in the scene (by default $L = 32$), and each layer $l \in \{1, ..., L\}$ is associated with a unique disparity value d_l. For each pixel p in $I_{0,0}$, we define its cost of being assigned to a specific layer l as:

$$C^l(p) = \sum_{(u,v)}^{I} ||I_{u,v}(p') - I_{0,0}(p)||,$$
$$p' = p + d_l(u, v). \tag{3}$$

Here, I is the image array representing the light field. Intuitively, $I_{0,0}(p')$ is the corresponding pixel of p in $I_{u,v}$

under the current disparity value d_l. If d_l is correct, then these two pixels should have similar colors, leading to a lower cost value $C^l(p)$. In this way we can construct a cost volume with axes (x, y, l), which is noisy given that the costs of each pixel are computed individually. Note that our cost function is different from the ones used in [38]. This is because in applications like optical flow or stereo matching, the cost function needs to be designed to be robust against illumination changes, while in our application the light field data has more consistent color values across the different views. Note that this cost function is defined over pixels instead of patches, given that a single patch on the depth boundary could contain pixels that have different disparity values. After the initial cost volume is computed, we follow the approach in [38] and smooth it by applying edge-aware guided filter [17] on the cost map C^l for each layer, where the radius of the filter is 9. Afterwords the layer label for each pixel $L(p)$ is calculated as:

$$L(p) = \arg\min_l C^l(p). \tag{4}$$

The initial labeling generates hard region boundaries. To achieve seamless compositing of different layers, we further process the regions to produce soft boundaries, by applying the guided image filter again on each layer with a small radius of 4. Examples of the soft layer mask extraction are shown in Figure 7. In the work of Zitnick et al. [56], an alpha matting method is used to extract soft layer masks in video. We also compare with a latest alpha matting method [42] for soft mask generation in Figure 7, which suggests that both methods produce visually similar results. However, the alpha matting method takes about 2-3 minutes to compute a soft mask for each layer, while guided image filtering takes only about 100-200 ms. We thus choose guided image filtering to produce soft layer masks in our implementation.

Colors at pixels with mask values between 0 and 1 are mixtures of object colors in different layers. We therefore apply color decontamination in each layer by extrapolating object colors from the interior of the object to its soft boundary region, under the assumption that object colors are spatially consistent. This is a common technique in image matting and compositing [49]. The extended layer after color decontamination is denoted as I^l, which serves as the basis for our patch-based synthesis process detailed in the next section.

5 Layered Synthesis with Patches

5.1 Why Layered Synthesis?

To demonstrate why layered synthesis is necessary in our application, we first consider a simple alternative approach. We can first synthesize a new central view image based on user inputs, then propagate edited pixels to all other views based on their disparities. As shown in the example in Figure 8, this strategy is likely to introduce artifacts near object boundaries. This is because different depth layers have different disparities, and a pixel at the depth boundary on the central view corresponds to different scene points in other views. In other words, modifying only the foremost layers in the central view image is insufficient for light field editing.

To avoid this problem our system employs a back-to-front layered synthesis framework for content consistency across views. To set up the context for explaining this framework, lets assume the user has specified a region ω on the central view image for an object removal task. Furthermore, for simplicity we assume the entire edited region belongs to depth layer l_ω. In practice if ω occupies multiple depth layers, we decompose it into sub-regions where each sub-region belongs to only a single depth layer, and apply the following procedures to all sub-regions sequentially. Given this user input, the depth layer l_ω and all others that are behind it are treated as layers that are affected by this operation, denoted as I^l, where $l = \{0, 1, ..., l_\omega\}$, in descending order of depth. For each affected layer I^l, we need to synthesize two things in order to complete it in ω: (1) the layer mask; and (2) the layer appearance. In other words, we need to recover a portion from region ω, which was originally occluded by a foreground object, and now occupied by a background layer, as well as the appearance of that portion. The only exception is the deepest layer I^0, for which we assume the layer mask includes the entire ω region to avoid creating holes in the final result.

The layered synthesis framework also allows us to measure the cross-view coherence defined in Equation (2) more

Fig. 6. Examples of soft layer mask extraction. Left: the central view of the light field. Right: the estimated depth map.

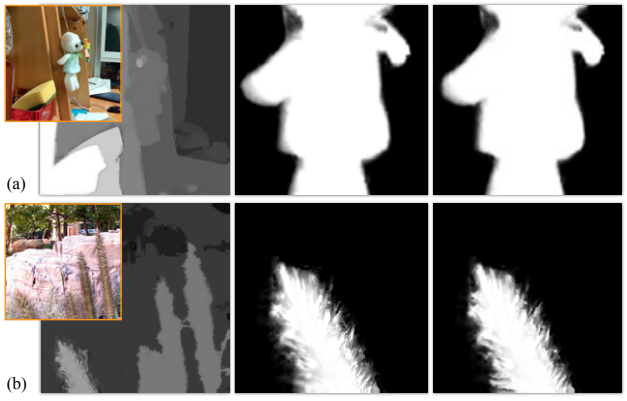


Fig. 7. Comparisons of soft layer mask generation. Left: the central view of the light field. Middle: Close-up soft masks of one layer, extracted by guided image filtering [17]. Right: Close-up soft mask extracted by an alpha matting method [42].

(a) (b) (c)

Fig. 8. The motivation for layered synthesis on the central view image. (a) original central view image; (b) edited central view as a regular image without layered synthesis; (c) propagate the result in (b) to view (-0.14, 0.24), notice the artifacts around the original object boundary. See another comparison to a non-layered synthesis result in the supplementary video.

easily. We can simply compute cross-view coherence in each edited layer individually without worrying about occlusion among different layers, and then sum the coherence values of different layers together as the overall coherence measure.

5.2 Layer Mask Synthesis

We first describe how to complete a layer mask in the user-specified region ω, which is essentially a shape completion task. This is done using a multi-scale patch-based synthesis approach in our system. We treat the layer mask outside ω as the source mask $\mathcal{M}_s$, which is known and fixed, and the entire layer mask including ω as the target mask $\mathcal{M}_t$ to be completed. We build Gaussian pyramids for both masks, denoted as $\mathcal{M}_s^{(0)}, \mathcal{M}_s^{(1)}, ..., \mathcal{M}_s^{(n)}$, and $\mathcal{M}_t^{(0)}, \mathcal{M}_t^{(1)}, ..., \mathcal{M}_t^{(n)}$, respectively. As shown in Figure 9, in order to fill in the hole $\omega^{(0)}$ in $\mathcal{M}_t^{(0)}$, we find the boundary of the layer mask in $\omega^{(0)}$ first. Specifically, we start from a pixel p that is on the hole boundary, and denote the patch centered at p as $N_t(p)$. p is on the hole boundary only if $N_t(p)$ contains both in-layer pixels ($p_i \in N_t(p)$ and $\mathcal{M}_t(p_i) = 1$) and off-layer ones ($p_j \in N_t(p)$ and $\mathcal{M}_t(p_j) = 0$). We then search in $\mathcal{M}_s^{(0)}$ to find a patch $N_s(q)$ that is most similar to $N_t(p)$, according to the distance metric defined as:

$$D_m(N_t(p), N_s(q)) = \sum_{i \in N} ||\mathcal{M}_s(i) - \mathcal{M}_t(i)||, \quad (5)$$

$\forall i$ where $\mathcal{M}_t(i)$ is known or already filled. This process is repeated until no hole boundary pixel remains. The remaining unfilled pixels are all inside the boundary and are filled with full opacity. This process is formally described in Algorithm. 1. Note that when the source mask boundaries are soft, this above process is capable of generating soft layer boundaries in ω as well.

Once the top level mask $\mathcal{M}_t^{(0)}$ has been completed, it is upsampled to the next level as an initialization for $\mathcal{M}_t^{(1)}$, which is then further refined by finding nearest patches in $\mathcal{M}_s^{(1)}$ according to Equation (5) and using them to update the target region $\omega^{(1)}$. This process continues until the bottom level of the pyramid is completed.

Our shape completion approach is similar to the shape synthesis method proposed by Rosenberger *et al.* [39] in the sense that both methods use hierarchical layers for shape synthesis. The difference is that they focus on local details and use a random initialization for the base layer and only then transfer the shape feature details in higher levels, while we need to synthesize a mask with the same structure as the original mask while satisfying boundary conditions, as shown in Figure 9.

Algorithm 1 Mask synthesis on $\mathcal{M}_t^{(0)}$

function SYNTHESIZE($\mathcal{M}_t^{(0)}$)
  while p = FindBoundaryPixel($\omega^{(0)}$) **do**
    if p is valid **then**
      $q_s \leftarrow \arg\min_q D_m(N_s(q), N_t(p)), q \in \mathcal{M}_s^{(0)}$
      for p_i in $N_s(p)$ **do**
        $p_i \leftarrow q_i$ in $N_t(q_s)$ if p_i is not filled
        Set p_i as filled pixel
      end for
    elsebreak;
    end if
  end while
  Fill pixels inside the boundary as 1.
  return
end function
function FINDBOUNDARYPIXEL($\omega^{(0)}$)
  for $p \in \omega^{(0)}$ *and* p *is not filled* **do**
    if $\exists p_i \in N_t(p)$ is filled and $\mathcal{M}_t^{(0)}(p_i) = 0$ and $\exists p_j \in N_t(p)$ is filled and $\mathcal{M}_t^{(0)}(p_j) > 0$ **then**
      return p
    end if
  end for
end function

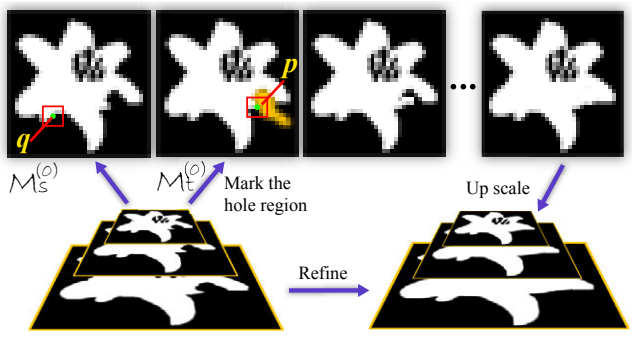


Fig. 9. Mask synthesis. After building the gaussian pyramids of the layer mask, the target region is defined by marking the hole region on the layer mask and the source region is taken as the rest of the layer mask. In the coarsest scale, the hole mask is filled using Algorithm. 1 and then up-scaled and refined using the corresponding source mask until the finest scale.

5.3 Layer Appearance Synthesis

The synthesized layer mask $\mathcal{M}_t$ defines the shape of the current layer in the target region, and we need to further synthesize its appearance for rendering purposes. The known region of the layer l outside ω is denoted as A_s^l, and the region to be synthesized (containing ω) is denoted as A_t^l. Our goal is to synthesize the appearance of ω so that A_t^l is consistent with A_s^l. This objective is mathematically defined as minimizing the following appearance distance between A_t^l and A_s^l:

$$D^l(A_t^l,A_s^l) = \sum_{p\in\mathcal{M}_t^+} \min_{q\in\mathcal{M}_s^+} D_a(N_t(p), N_s(q)), \quad (6)$$

where $\mathcal{M}_t^+$ includes all pixel locations i so that $\mathcal{M}_t(i) > 0$. Similarly, $\mathcal{M}_s^+$ refers to the non-zero region in $\mathcal{M}_s$. p and q are pixel indexes in A_t^l and A_s^l, respectively, and $N_t(p)$ and $N_s(q)$ are local image patches that center at them. $D_a(\cdot)$ is the appearance distance metric between two image patches, which plays the central role in enforcing appearance coherence. We thus can rewrite Equation (1) using the layered representation:

$$E_f(u,v) = \sum_{l=1,\ldots,L} D^l(A_t^l, A_s^l), \quad (7)$$

The simplest way to define $D_a(\cdot)$ is to define it as the sum of Euclidean distances of corresponding pixel colors. In our application, considering that the layer appearance could be spatially-varying towards the boundary of the region, we prefer to use patches that are on the layer mask boundary in A_s^l to synthesize boundary patches in ω in A_t^l. To achieve this we also include the mask similarity in the patch appearance distance metric, which is defined as:

$$D_a(N_t(p),N_s(q)) = \alpha||N_s(q) - N_t(p)|| + (1-\alpha)D_m(N_s(q), N_t(p)), \quad (8)$$

where the balancing weight α is set at 0.5 in the system. We solve this optimization problem using the multi-scale patch synthesis approach [53] with the modified distance metric. The target region in the coarsest level is initialized with random offset values. In the finer levels, the offsets are initialized by upsampling the offsets in the previous level. Given the offsets map, the ω region in A_t^l are synthesized using the same pixel voting method as in [53].

In our experiments, we use 5×5 patches and set the default number of pyramid levels to four (image size is 68×68 at the coarsest level). At each level, we perform four PatchMatch (propagation and random search) iterations [2]. Because the masks provide extra constraints during the patch search and the search space is limited to the same layer, a few EM iterations are sufficient for a good convergence - we perform three iterations at the coarsest scale and two at finer scales. An example of the final synthesis result is shown in Figure 10. In this way, the energy in Equation (7) is minimized by assigning the optimized color value for every pixel in each layer we extracted.

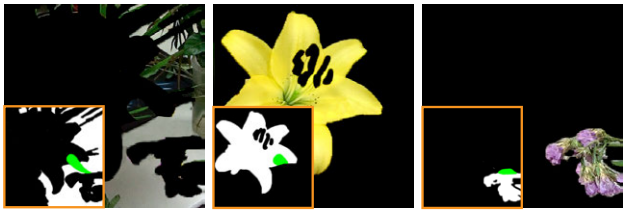


Fig. 10. Layered synthesis results of the example shown in Figure 2. In each layer, the green region in the mask is the target region to fill.

6 Edit Propagation Across Views

After the central view image has been edited, we propagate the edits to all the other views while keeping the appearance consistency and the perceived depth.

6.1 Initial Propagation

For a layer l in the central view $I_{0,0}$, the user-specified edits may produce two regions: one caused by object removal (i.e. the hole), denoted as $R_{0,0}^l$; the other associated with object insertion (i.e. synthesized new object), denoted as $R_{0,0}'^l$. The goal of edit propagation is to first find the corresponding regions $R_{u,v}^l$ and $R_{u,v}'^l$ in another view $I_{u,v}$, and then accomplish two tasks: (1) synthesize a new object in $R_{u,v}'^l$; and (2) fill the remaining hole $\{\bar{R}_{u,v}^l | \bar{R}_{u,v}^l = R_{u,v}^l - R_{u,v}'^l\}$, as shown in Figure 11.

Color-based propagation. $R_{u,v}^l$ and $R_{u,v}'^l$ can be found using the disparity value d_l. A straightforward way to accomplish the above two tasks is: for a pixel p that needs to be synthesized in $I_{u,v}$, find its corresponding pixel location in $I_{0,0}$. Then find the topmost layer at this location that is visible by examining the layer masks, and copy the pixel value on this layer to p. For the example shown in Figure

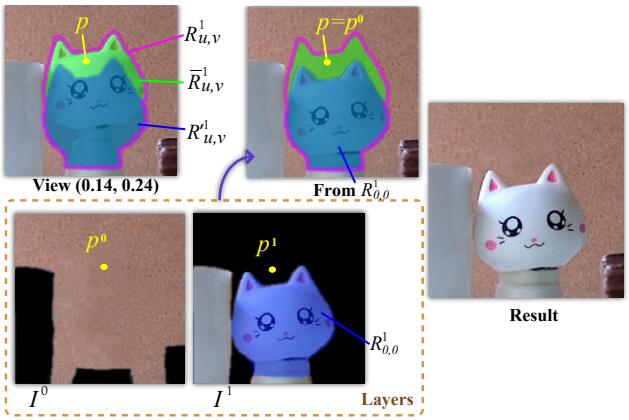


Fig. 11. Edit propagation from the central view (dashed bounding box) to another view. For pixel p in $\bar{R}^1_{u,v}$ (top-left), p^0 and p^1 are the corresponding pixels computed using the disparity in layer I^0 and I^1, respectively (bottom-left). Since $\mathcal{M}^0_{p^0} > 0$ and $\mathcal{M}^1_{p^1} = 0$, the color of p^0 is assigned to p. The final propagated result of the new view is on the right.

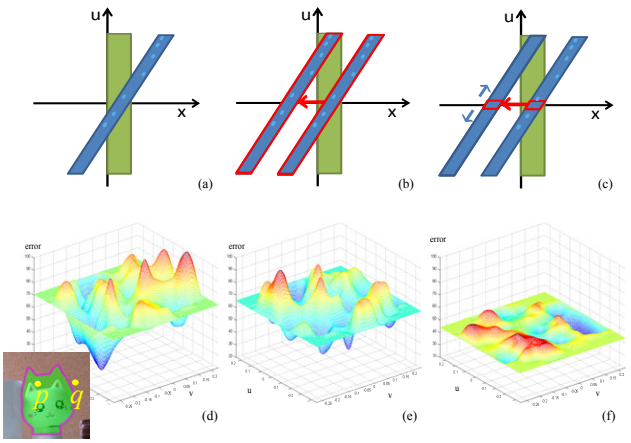


Fig. 12. Comparing color-based and offset-based editing propagation. (a-c) The schematic diagrams of 2D-slices for: original light field, edited results using patch-offset-based propagation, and results using direct color-based propagation. (d) The SSD patch distance map for patches centered at q, an un-edited pixel in the original data (a real example, shown in the lower left corner). (e) The SSD distance map for patches centered at p, a pixel edited by offset-based propagation. (f) The distance map for p by color-based propagation. Note that (d) and (e) contain similar variations while (f) is too flat.

11, the corresponding pixel location of p is not visible on the front layer (layer 1), so the pixel value on layer 0 (denoted as p^0) is assigned to p.

The above method can generate plausible results, but one drawback is that all corresponding pixels across different views will have exactly the same synthesized color. In contrast, in the original light field, the same scene point often has a small appearance variance across views, as the object surfaces are rarely pure Lambertian. In addition, to retain a natural image appearance we want to preserve the original image noise. Such an example is shown in Figure 12. Lets consider two pixels p and q on $I_{0,0}$, where the former is inside the user-edited region and consequently is assigned to a new color across all views, and the latter is outside the edited region, and thus is unchanged. If we take patches centered at them in different views, and compute the SSD patch distances, we can see for q the typical patch distances have relatively large variations that naturally exist in the light field data (Figure 12d), while for p the variations are much smaller (Figure 12f).

Offset-based propagation. To make the synthesized regions look more realistic, we do not directly copy the pixel values from $I_{0,0}$ to other views. Instead, for pixel p that is edited in $I_{0,0}$, we look at the location of source patch $S(p)$ that was chosen during the patch based synthesis process (the offset value from the nearest neighbor field in the last EM iteration, see Section 5.3). We then find the corresponding source location in $I_{u,v}$, and use its color for p in the edited region of $I_{u,v}$. In other words, our propagation process propagates *patch offsets* to ensure: (1) $I_{u,v}$ is edited using pixel colors sampled from the image itself; and (2) the edits applied in $I_{u,v}$ are consistent with the ones in $I_{0,0}$ as well as in other views. It essentially uses 4D-chunks in the original light field to generated the novel content.

Figure 12 shows a comparison of color-based and offset-based propagation methods, on their abilities to maintain the original object appearance variation across views. It suggests that the offset-based propagation generates more natural appearance variations that are consistent with the original data. We thus choose the offset-based propagation scheme to generate final results. With better cross-view coherence when assigning pixel values, the cost function defined by Equation (2) is also minimized. Next, we describe a sub-pixel refinement method which further reduces the cost.

6.2 Sub-pixel Refinement

Typical light field data contains evenly distributed views (Figure 3), where the scene content changes smoothly across neighboring views. Thus ideally, the edited light field should maintain the same level of cross-view smoothness. However the initial editing propagation process does not consider sub-pixel accuracy, thus small alignment errors may accumulate and cause noticeable sudden scene change across certain neighboring views.

Our system adopts an optical-flow-based sub-pixel refinement approach to eliminate such artifacts, based on the observation that optical flow is capable of detecting the actual region shifts in the neighboring views. Specifically, for each edited layer of the current view i in the edited light field, we compute the optical flow using the method in [46] from the same layer of all neighboring views j to the current view i, denoted as F_{ji}, and compute the sub-pixel

shift for the current layer in view i as:

$$\delta_i = -\frac{1}{N}\sum_{j}^{N} F_{ji}, \tag{9}$$

where N is the number of neighboring views. This is essentially applying a local smoothing on the region shift vectors across views.

We iteratively apply the correction for all the views and update them if necessary, until all corrections are less than a small threshold δ (set at $(0.2, 0.2)$ in our system). The effect of this sub-pixel refinement step is shown in the supplementary video. In our experiments, the refinement will converge in 2-3 iterations.

7 Applications and Results

We have successfully applied the proposed approach in a wide range of plenoptic image editing applications, such as region completion, reshuffling, parallax magnification and depth manipulation. Due to limited space we only show a few examples here. Given the difficulty of showing the light field in still figures, we adopt a visualization method of showing outputs with different focus planes, to illustrate that our results contain the correct depth. Complete results can be seen in the supplementary video.

The complete light field data, both original and synthesized and an executable for viewing such data, are all included in the supplementary materials. For the ease of comparison with the original light field, we chose to use the same hexagonal layout with 37 views as our outputs. However technically, our system is not limited to it and we can render any in-between views using common view interpolation techniques. Unless specified, all the light field data are captured using a Lytro camera. We use Lytro's desktop software to extract the rendered image arrays containing 37 views from the LFP files exported by the Lytro camera, and use them as the inputs to our system.

7.1 Completion and Reshuffling/Cloning

To remove an object in the input light field, we mark pixels in all affected layers in the user specified region as "hole pixels", and complete each layer using the layered synthesis process described in Section 5.

After all affected layers are completed, the entire light field is modified using the propagation method described in Section 6. One completion result is shown in Figure 13(a). A reshuffling or cloning an image region to a target position is implemented based on the completion algorithm above. To move an object to another place, we first use the completion method to fill in the region of the original object. By adding hard constraints in the target region of the patch offset map, a new object can be directly synthesized at the target location using the algorithms in Section 5. Reshuffling example is shown in Figure 13(b), more examples are in the supplementary materials.

7.2 Changing Depth

The proposed method can be used to change the depth of an object in a light field. This is accomplished in two steps: we first remove the user-specified object from its original layer using hole filling. Then the object and its layer mask are copied onto the new depth layer specified by the user as an initialization, followed by a similar patch-based synthesis process to blend it into the target depth layer. An example is shown in Figure 13(c). We can also combine multiple operations, such as depth changing and reshuffling, to achieve more sophisticated editing goals, as shown in Figure 13(d).

7.3 Parallax Magnification

One of the major limitations of existing consumer light field cameras is their narrow viewpoint range, especially when the scene is far from the camera. Our system enables a new important application of parallax magnification, by synthesizing the appearance of occluded pixels in each layer. Specifically, given a magnifying factor the user wants to apply to the viewpoint range, we first compute the synthesis region in each layer of the central view M^l that will be revealed in the output views. We denote the inverted mask $\tilde{M}^l = 1 - M^l$ as the region that needs to be synthesized for this application. Specifically, the maximum region we need to synthesize is the part in $\tilde{M}^l$ that will be revealed, i.e. cannot be covered by all layers in front of l, in the farthest perspectives from the central view. It is computed by:

$$R^l = \bigcup_{j \in S^l} ((\tilde{M}^l - M_1^j) \bigcup (\tilde{M}^l - M_2^j)), \tag{10}$$

where M_1^j and M_2^j are the shifted masks for layer j by the vector of (u_{max}, v_{max}) and (u_{min}, v_{min}) of the generated novel views, i.e., the largest disparity vectors. S^l denotes the set of layers that is in front of l. $(\tilde{M}^l - M_x^j)$ means subtracting pixels belonging to M_x^j from $\tilde{M}^l$. By completing each layer in R^l and recovering the entire light field according to the given new camera coordinates, we can magnify the parallax in the original light field. This is shown in Figure 14(c).

In this application, the regions that need to be synthesized are larger if the user wants a wider range of viewpoints. Larger "holes" are notoriously harder to fill automatically so previous patch-based methods employed user constraints to obtain good results [2]. Similarly, we allow the user to specify structural constraints that are used to locally limit the search region of the marked pixels (see [2] for more details). We include a comparison between results with and without constraints in our supplemental video. To evaluate our method, we also render two light fields from a CG scene with different parallax range. Figure. 14(d) shows the left and right most views generated by our method and the real corresponding image rendered from that scene.

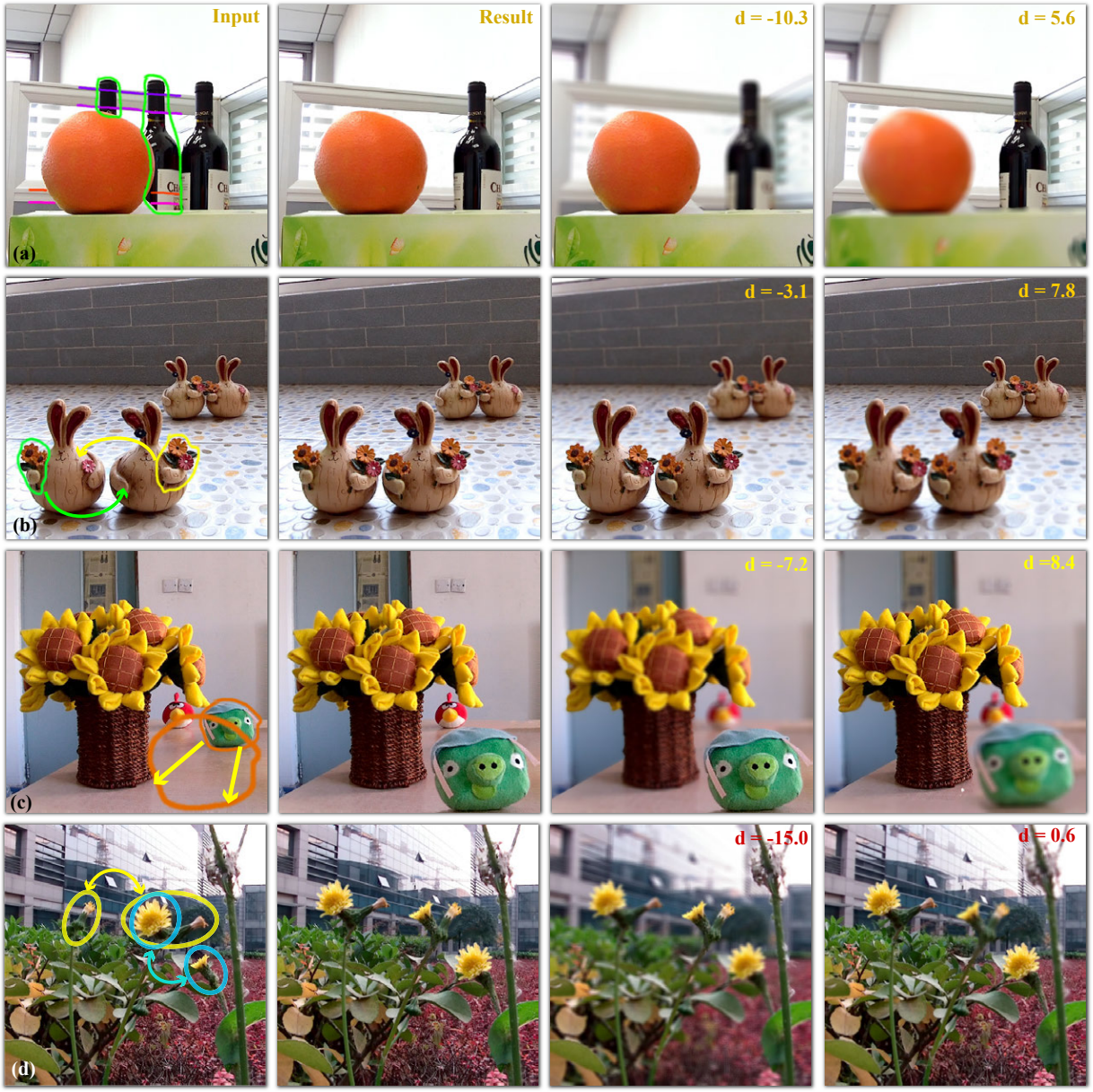


Fig. 13. Applying our system to different light field editing tasks. (a) Removing selected objects. (b) Reshuffling the selected leaf region. (c) Changing object depth. (d) Combining reshuffling with a depth change. The third and fourth columns are refocusing results to show that our results contain the correct depth. The complete results are shown in the supplementary video.

7.4 Light Field Super-resolution

Due to hardware limitations, the resolution of current consumer-level plenoptic cameras is much lower than the resolution of DSLRs. Using our framework, we can increase the resolution of the light-field, with the help of a reference high resolution image taken by a regular camera from the same perspective, denoted as I_r.

We first increase the resolution of each layer in the central view $I_{0,0}$ using I_r. On each layer l, we compute the region R_u^l that we need to up-scale in a similar way to Equation (10) (with the magnification rate 1). Since this region contains pixels that cannot be seen in $I_{0,0}$, we fill the holes using corresponding visible pixels from other views. Using the non-dyadic filter banks proposed in [13], we downsize I_r to the size of $I_{0,0}$ to get the low frequency and high frequency bands for each patch, and up-scale I_u^l to the size of I_r. In I_r, we search for the patch whose lower frequency band is most similar to the patch in up-scaled $I_{0,0}$. We then use the corresponding high resolution patches to generate the up-scaled I_u^l. In our experiment, the resolution of the reference picture is 3 times of the resolution of $I_{0,0}$, and we use 5:4, 5:4, 4:3, and 3:2 filter banks as in [13]. Using a similar process to Section 6, the nearest neighbor field is used to generate an up-scaled image for each view.

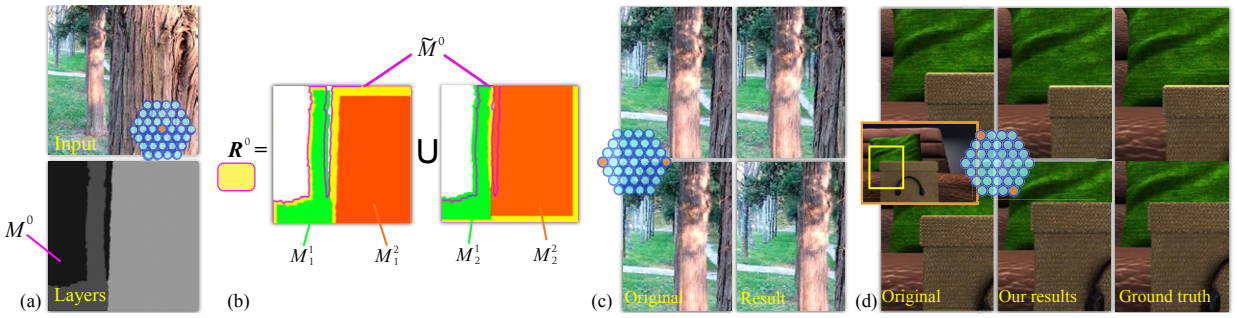


Fig. 14. Parallax magnification. (a) The central view and its depth layers; (b) The region needs to be synthesized (yellow) for the background layer (layer 0) computed from the farthest views (see Section 7.3); (c) Top row: the original leftmost and rightmost views. Bottom row: enlarged parallax of the leftmost and rightmost views by a factor of 8. (d) A comparison with the ground truth of a synthetic light field.

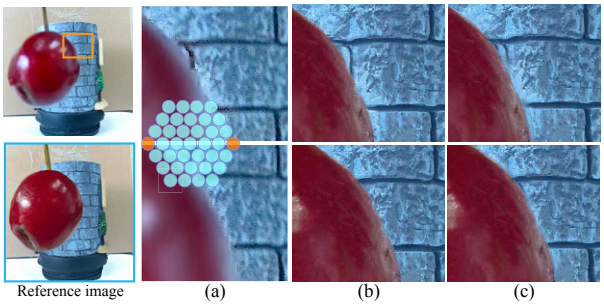


Fig. 15. Light field upscaling. (a) Bicubic upsampling of two different views. (b) Results of our method. (c) Results of [5].

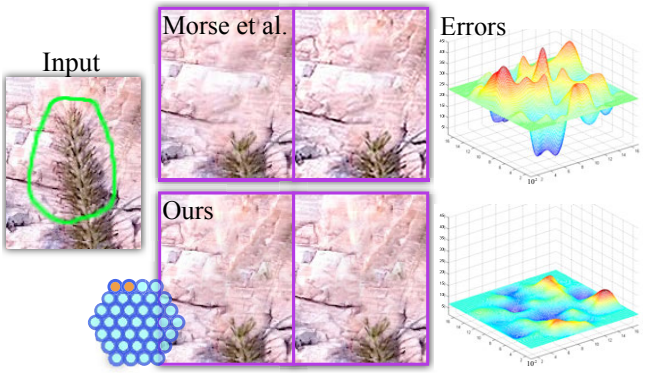


Fig. 16. Comparison with the stereo inpainting method of Morse et al. Left: input central view with a user specified hole. Right: two neighboring views in the completed results using the method of Morse et al. (top) and our approach (bottom). The error map shows the sum of content difference between all other views and the central view. Our method generates much smaller errors.

Figure. 15 shows one super-resolution example. We compare our result with the recent patch-based method of Boominathan *et al.* [5]. Their method uses a uniform patch-based upsampling approach for all pixels, ignoring boundary discontinuities. Our method generates more consistent results along layer boundaries. See supplemental video for further examples.

8 Discussion

8.1 More Comparisons

Morse *et al.* [34] use a patch-based synthesis method to complete stereo image pairs. They first estimate the disparities in the hole, then compute the nearest neighborhood fields (NNF) by considering both appearance and depth consistency and allowing propagation between different views. Then a stereo-consistent patch blending step is applied to produce the final result. One may argue that this method could be potentially extended to light field editing.

However, when adapting this method to light field completion, the random nature of the PatchMatch algorithm [2] used in that method, makes it unlikely to complete a region so that it appears consistent in different views (even though the consistency has been considered when computing NNFs and blending patches). In Figure. 16 we extend this method from handling stereo pairs to a plenoptic image, by simultaneously optimizing all the views in the light field in the hole region. When generating the final image, we generate the blending weights for each patch by taking into account all SSD-distances of corresponding patches in other views (extension of their weighting function to multiple images).

To check the content consistency across different views, we visualize the difference map of the filled region between each view and the central view in Figure. 16. As can be seen, the difference map produced by the generalization of their method contains large pixel-wise difference, indicating content jittering of their results when shifting perspectives. This kind of jitter is not an issue for perceiving stereo but is a major artifact for viewing a plenoptic image. Please refer to the supplementary video for the complete comparison.

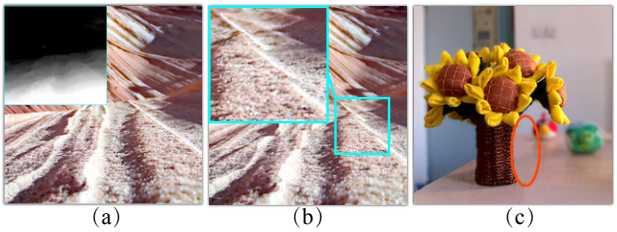
8.2 Run-time

The light field data used in our experiments was captured by a Lytro camera. Images of different views were resized to 544×544. We implemented our approach in C++ on a PC with an Intel Xeon E5620 CPU with 4 cores (8 threads) at 2.4GHz and 8GB RAM. In our experiments, using a single

core, extracting the depth layers takes about 1 second, the synthesis on each related layer takes about $0.5 \sim 3$ seconds, and it costs about $3 \sim 6$ seconds to propagate the edits on each layer to the entire light field. Some components of our method can be easily parallelized, like the cost-volume filtering in the depth layers extraction and edit propagations across views. The synthesis of each layer can be also trivially parallelized. It takes about $0.2 \sim 0.4$ seconds to extract the depth layers and $0.5 \sim 0.8$ seconds to propagate edits to all views. The overall run-time using a parallel implementation for all the examples in the paper was between 3 to 10 seconds (e.g., the examples from Fig. 1 and Fig. 13(a) took 3.7 and 6.3 seconds, respectively).

8.3 Limitations

The proposed approach does have several limitations. First, when the captured scene contains a large ground plane or other non-frontal surface with a smoothly-varying depth, as the example shown in Figure 17(a), our method might not work well. The slanted surface cannot be well modeled as a single depth layer, so applications like completion and reshuffling might exhibit artifacts, especially if the surface has visually salient patterns. Furthermore, since we do not take into account any relation between neighboring layers in our representation, scenes with contiguous dominant structures which span across multiple layers might cause artifacts in applications like parallax widening. Figure 17(b) shows such an example, where the trail on the ground cannot be preserved as a continuous line in the farthest view, and instead is decomposed into discrete segments contained in different layers. However, our method often works well for slanted planes that do not contain salient patterns or strong textures (e.g. in Figure 17(c)).



(a) (b) (c)

Fig. 17. Our system does not work well if the scene contains visually salient patterns in a plane with smoothly-varying depth. (a) Central view of the input, which contains a large smoothly-varying depth layer; (b) Synthesized farthest view with a widened parallax where the long line structure is broken. (c) Our method can still generate fine results for slanted surfaces that are not highly textured.

In addition, our system requires a rather accurate depth layer extraction for high quality results. However accurate automatic depth estimation is a challenging computer vision problem, especially for complex scenes, even in a structured setting such as a lightfield camera. In order to obtain more realistic results and a better consistency of the perceived depth, our system allows the user to manually specify additional constraints as shown in Figure. 13(a), and to adjust the depth values when propagating edits across different views. Out of all the results shown in the paper, only the examples in Figure. 13(a) were obtained using manual adjustments.

Finally, our current system extracts soft layer masks using guided image filtering with a small radius. It works well for objects with near-solid or slightly furry boundaries, but cannot handle large semi-transparent regions such as long hair blowing in the wind. Additional user input is needed to specify such regions, and more advanced alpha matting methods can be potentially employed for creating more accurate soft masks in such cases.

9 Conclusion and Future Work

We have demonstrated an interactive system for light field editing using recent advances in patch-based image synthesis methods. Our method represents the light field as an image set with a hexagonal layout, and models the captured scene as overlapping layers with different depths. After the user specifies an editing task in the central view image, the method employs a back-to-front layered synthesis to edit all affected layers. The edits on these layers are then propagated to all other views while maintaining the appearance consistency and the perceived depth. Experimental results show that our method allows applications like completion, reshuffling and parallax magnification to be performed effectively on plenoptic images for the first time. Additional results are shown in the supplementary materials.

As future work, we plan to consider additional geometric features and inter-layer correlations in the layered scene representation, to handle more complex scenes such as the one shown in Figure 17. We also plan to extend this framework to enable new applications such as plenoptic image compositing and merging.

Acknowledgments

This work was supported by the National High Technology Research and Development Program of China (Project No. 2013AA013903), the Natural Science Foundation of China (Project No. 61521002, 61272226), Research Grant of Beijing Higher Institution Engineering Research Center, the General Financial Grant from the China Postdoctoral Science Foundation (Grant No. 2015M580100), and Tsinghua University Initiative Scientific Research Program.

References

[1] Adobe, "Photoshop cc content-aware fill," 2013, http://www.adobe.com/technology/projects/content-aware-fill.html.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," *ACM TOG*, vol. 28, no. 3, pp. 24:1–24:11, Jul. 2009.

[3] C. Birklbauer and O. Bimber, "Light-field retargeting," in *Computer Graphics Forum*, vol. 31, no. 2pt1, 2012, pp. 295–303.

[4] T. E. Bishop, S. Zanetti, and P. Favaro, "Light field superresolution," in *Proc. ICCP*, 2009, pp. 1–9.

[5] V. Boominathan, K. Mitra, and A. Veeraraghavan, "Improving resolution and depth-of-field of light field cameras using a hybrid imaging system," in *Proc. ICCP*, 2014.

[6] B. Chen, E. Ofek, H.-Y. Shum, and M. Levoy, "Interactive deformation of light fields," in *Proc. I3D*. ACM, 2005, pp. 139–146.

[7] D. G. Dansereau, O. Pizarro, and S. B. Williams, "Decoding, calibration and rectification for lenselet-based plenoptic cameras," in *Proc. CVPR*, 2013.

[8] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining inconsistent images using patch-based synthesis," *ACM TOG*, vol. 31, no. 4, pp. 82:1–82:10, Jul. 2012.

[9] W. Dong, F. Wu, Y. Kong, X. Mei, T.-Y. Lee, and X. Zhang, "Image retargeting by texture-aware synthesis," *IEEE TVCG*, vol. 22, no. 2, pp. 1088–1101, Feb 2016.

[10] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. SIGGRAPH*, 2001, pp. 341–346.

[11] K. Egan, F. Hecht, F. Durand, and R. Ramamoorthi, "Frequency analysis and sheared filtering for shadow light fields of complex occluders," *ACM TOG*, vol. 30, no. 2, pp. 1–13, Apr. 2011.

[12] A. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image-based rendering using image-based priors," in *Proc. ICCV*. Washington, DC, USA: IEEE Computer Society, 2003, p. 1176.

[13] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM TOG*, vol. 30, no. 2, p. 12, 2011.

[14] M. Fuchs, M. Kächele, and S. Rusinkiewicz, "Design and fabrication of faceted mirror arrays for light field capture," *Computer Graphics Forum*, vol. 32, no. 8, pp. 246–257, Aug. 2013.

[15] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. CVPR*, 2009.

[16] B. Goldluecke and S. Wanner, "The variational structure of disparity and regularization of 4d light fields," in *IEEE CVPR*, June 2013, pp. 1003–1010.

[17] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE TPAMI*, vol. 35, no. 6, pp. 1397–1409, 2013.

[18] M. Holroyd, I. Baran, J. Lawrence, and W. Matusik, "Computing and fabricating multilayer models," *ACM TOG*, vol. 30, no. 6, pp. 187:1–9, 2011.

[19] D. R. Horn and B. Chen, "Lightshop: interactive light field manipulation and rendering," in *Proc. I3D*, 2007, pp. 121–128.

[20] Y. Itoh and G. Klinker, "Light-field correction for spatial calibration of optical see-through head-mounted displays," *IEEE TVCG*, vol. 21, no. 4, pp. 471–480, 2015.

[21] A. Jarabo, B. Masia, A. Bousseau, F. Pellacini, and D. Gutierrez, "How do people edit light fields?" *ACM TOG(SIGGRAPH 2014)*, vol. 33, no. 4, 2014.

[22] A. Jarabo, B. Masia, and D. Gutierrez, "Efficient propagation of light field edits," in *Proc. of SIACG*, 2011.

[23] C. Kim, A. Hornung, S. Heinzle, W. Matusik, and M. Gross, "Multi-perspective stereoscopy from light fields," *ACM TOG*, vol. 30, no. 6, p. 190, 2011.

[24] K. Kodama and A. Kubota, "Efficient reconstruction of all-in-focus images through shifted pinholes from multi-focus images for dense light field synthesis and rendering," *IEEE Transactions on Image Processing*, vol. 22, no. 11, pp. 4407–4421, 2013.

[25] D. Lanman and D. Luebke, "Near-eye light field displays," *ACM TOG*, vol. 32, no. 6, pp. 220:1–220:10, 2013.

[26] J. Lehtinen, T. Aila, J. Chen, S. Laine, and F. Durand, "Temporal light field reconstruction for rendering distribution effects," *ACM TOG*, vol. 30, no. 4, pp. 55:1–55:12, Jul 2011.

[27] A. Levin and F. Durand, "Linear view synthesis using a dimensionality gap light field prior," in *Proc. CVPR*. IEEE, 2010, pp. 1831–1838.

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

[29] M. Levoy, Z. Zhang, and I. McDowall, "Recording and controlling the 4d light field in a microscope using microlens arrays," *Journal of Microscopy*, vol. 235, no. 2, pp. 144–162, 2009.

[30] Y. Liu, L. Sun, and S. Yang, "A retargeting method for stereoscopic 3d video," *Computational Visual Media*, vol. 1, no. 2, pp. 119–127, 2015.

[31] W.-Y. Lo, J. van Baar, C. Knaus, M. Zwicker, and M. Gross, "Stereoscopic 3d copy & paste," *ACM TOG*, vol. 29, no. 6, pp. 147:1–147:10, 2010.

[32] H. Lu, R. Pacanowski, and X. Granier, "Position-dependent importance sampling of light field luminaires," *IEEE TVCG*, vol. 21, no. 2, pp. 241–251, Feb 2015.

[33] S.-J. Luo, I.-C. Shen, B.-Y. Chen, W.-H. Cheng, and Y.-Y. Chuang, "Perspective-aware warping for seamless stereoscopic image cloning," *ACM TOG*, vol. 31, no. 6, pp. 182:1–182:8, Nov. 2012.

[34] B. Morse, J. Howard, S. Cohen, and B. Price, "Patchmatch-based content completion of stereo image pairs," in *Proc. 3DIMPVT*, 2012, pp. 555–562.

[35] R. Ng, "Fourier slice photography," *ACM TOG*, vol. 24, no. 3, pp. 735–744, 2005.

[36] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report*, vol. 2, no. 11, 2005.

[37] Y. Niu, W.-C. Feng, and F. Liu, "Enabling warping on stereoscopic images," *ACM TOG*, vol. 31, no. 6, pp. 183:1–183:10, 2012.

[38] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *Proc. CVPR*. IEEE, 2011, pp. 3017–3024.

[39] A. Rosenberger, D. Cohen-Or, and D. Lischinski, "Layered shape synthesis: Automatic generation of control maps for non-stationary textures," *ACM TOG*, vol. 28, no. 5, pp. 107:1–107:9, Dec. 2009.

[40] S. M. Seitz and K. N. Kutulakos, "Plenoptic image editing," *IJCV*, vol. 48, no. 2, pp. 115–129, 2002.

[41] P. Sen, N. K. Kalantari, M. Yaesoubi, S. Darabi, D. B. Goldman, and E. Shechtman, "Robust Patch-Based HDR Reconstruction of Dynamic Scenes," *ACM TOG (SIGGRAPH Asia 2012)*, vol. 31, no. 6, 2012.

[42] E. Shahrian and D. Rajan, "Weighted color and texture sample selection for image matting," in *IEEE CVPR*, 2012, pp. 718–725.

[43] E. Shechtman, A. Rav-Acha, M. Irani, and S. Seitz, "Regenerative morphing," in *Proc. CVPR*, San-Francisco, CA, June 2010.

[44] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand, "Light field reconstruction using sparsity in the continuous fourier domain," *ACM TOG*, vol. 34, no. 1, p. 12, 2014.

[45] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala, "Light field video stabilization," in *Proc. ICCV*. IEEE, 2009, pp. 341–348.

[46] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *Proc. CVPR*. IEEE, 2010, pp. 2432–2439.

[47] R.-F. Tong, Y. Zhang, and K.-L. Cheng, "Stereopasting: Interactive composition in stereoscopic images," *IEEE TVCG*, vol. 19, no. 8, pp. 1375–1385, 2013.

[48] K. Venkataraman, D. Lelescu, J. Duparr, A. McMahon, G. Molina, P. Chatterjee, R. Mullis, and S. Nayar, "Picam: an ultra-thin high performance monolithic camera array," *ACM TOG*, vol. 32, no. 6, pp. 166:1–166:12, 2013.

[49] J. Wang and M. Cohen, "Image and video matting: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 2, pp. 97–175, 2007.

[50] L. Wang, S. Lin, S. Lee, B. Guo, and H.-Y. Shum, "Light field morphing using 2d features," *IEEE TVCG*, vol. 11, no. 1, pp. 25–34, 2005.

[51] S. Wanner, C. Straehle, and B. Goldluecke, "Globally consistent multi-label assignment on the ray space of 4d light fields," in *IEEE CVPR*, 2013, pp. 1011–1018.

[52] G. Wetzstein, D. Lanman, M. Hirsch, and R. Raskar, "Tensor Displays: Compressive Light Field Synthesis using Multilayer Displays with Directional Backlighting," *ACM TOG*, vol. 31, no. 4, pp. 1–11, 2012.

[53] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE TPAMI*, vol. 29, no. 3, pp. 463–476, Mar. 2007.

[54] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging using large camera arrays," *ACM TOG*, vol. 24, no. 3, pp. 765–776, 2005.

[55] Z. Zhang, L. Wang, B. Guo, and H.-Y. Shum, "Feature-based light field morphing," *ACM TOG*, vol. 21, no. 3, pp. 457–464, 2002.

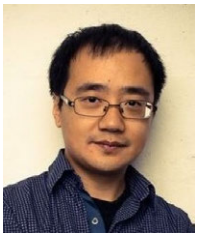
[56] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM TOG*, vol. 23, no. 3, pp. 600–608, Aug. 2004.

Eli Shechtman is a Senior Research Scientist at Adobe Research. He received the B.Sc. degree in electrical engineering (magna cum laude) from Tel-Aviv University in 1996. Between 2001 and 2007 he attended the Weizmann Institute of Science where he received with honors his M.Sc. and Ph.D. degrees. He was awarded the Weizmann Institute Dean prize for M.Sc. students, the J.F. Kennedy award (highest award at the Weizmann Institute) and the Knesset (Israeli parliament) outstanding student award. He received the best paper award at ECCV 2002 and a best poster award at CVPR 2004. His research interests include image and video processing, computational photography, object recognition and patch-based analysis and synthesis. He is a member of the IEEE and the ACM.

Zi-Ye Zhou received his BS degree from the Tsinghua University in 2014. He is currently a Phd candidate in University of Pennsylvania. His research interests include computer graphics, image processing and enhancement.

Fang-Lue Zhang received his Ph.D (2015) degree from Department of Computer Science from Tsinghua University and BS degree from the Zhejiang University in 2009. He is currently a post-doctor in Tsinghua University. His research interests include computer graphics, image processing and enhancement, image and video analysis and computer vision. He is a member of IEEE and ACM.

Jia-Xin Shi is an undergraduate student at the Tsinghua University. His research interests include computer graphics, image processing and enhancement and machine learning.



Jue Wang is a Principle Research Scientist at Adobe Research. He received his B.E. (2000) and M.Sc. (2003) from Department of Automation, Tsinghua University, Beijing, China, and his Ph.D (2007) in Electrical Engineering from the University of Washington, Seattle, WA, USA. He received Microsoft Research Fellowship and Yang Research Award from University of Washington in 2006. He joined Adobe Research in 2007 as a research scientist. His research interests include image and video processing, computational photography, computer graphics and vision. He is a senior member of IEEE and a member of ACM.

Shi-Min Hu is currently a professor in the department of Computer Science and Technology, Tsinghua University, Beijing. He received the PhD degree from Zhejiang University in 1996. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conference. He is Editor-in-Chief of Computational Visual media, and on editorial board of several journals, including IEEE Transactions on Visualization and Computer Graphics, Computer Aided Design and Computer & Graphics.