Efficient Propagation of Light Field Edits

Adrian Jarabo, Belen Masia and Diego Gutierrez

Universidad de Zaragoza

ABSTRACT

Light field editing is a complex task, due to the large amount of data and the need to keep consistency between views. This has hampered the creation of efficient edit propagation methods, similar to those existing for single images. We propose a framework to edit light fields at interactive rates, by propagating some sparse user edits in the full light field. This propagation is guided by a novel affinity function, which forces similar pixels (defined by our affinity space) to receive similar edits, thus ensuring consistency. To manage the light field's large amount of data, we propose a novel multi-dimensional downsampling technique: we first cluster pixels with high affinity, and then perform edit propagation over the downsampled data. We finally upsample back to the original full resolution, maintaining visual fidelity and view consistency between views.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Light fields [LH96, GGSC96] allow photo-realistic rendering of real objects and scenes independently of the complexity of its geometry and reflectance. They capture the appearance of objects using photographs, which map the incoming radiance in a four-dimensional parametrization of rays in free-space. Several works have been published regarding both the acquisition and rendering of light fields, finding good applications in videogames [HC07], augmented reality [CNR08] or novel computational photography techniques [Ng05, VRA*07] to refocus or reduce noise. However, manipulation and edition of light fields has received very little attention in the existing literature.

Editing the appearance of a light field is challenging because of some inehrent difficulties. First, preserving the consistency between samples is mandatory to maintain realism. Additionally, the amount of data to manipulate is usually very large, since the size of the light fields tends to be very high in order to provide well-sampled data. Managing such data sizes can make the editing process very slow, which is unacceptable if we are aiming to provide an interactive editing process to the user.

Recently, image editing methods based on sparse strokes-based edit propagation have allowed the user to perform complex edits by defining a few coarse strokes, and then propagating them to the rest of the image. These approaches are usually guided by the principle that similar pixels should receive similar edits. We base our novel efficient edit light field propagation technique on an extension of one of those methods, namely AppProp [AP08]. Our light field editing framework succeeds in solving the difficulties stated before: it provides a solution which maintains the coherence between the samples in the light field, by taking advantage of the fact that coherent elements in the light field should receive similar edits, as far as they have similar appearance; and it is done very efficiently, giving user feedback in interactive times, by downsampling the light field data such that similar and close pixels are clustered together.

In order to make this possible, we propose in this work the following contributions:

- A new similarity metric, adapted to the context of light field editing, which is defined to model the affinity between pixels, to guide the edit propagation process.
- A novel, multidimensional downsampling-upsampling algorithm, based on the affinity between elements in a light field, to provide fast and accurate affinity-based edit propagation.
- A framework for efficient light field editing, based on recent sparse strokes-based edit propagation methods, which allows interactive light field editing for the first time.

2. Related work

Few works have been developed in the context of efficient light field editing and manipulation. One of the first approaches is the plenoptic image editing work [SK02], which propagates edits from one sample to others by using a voxel-representation of the light field. This system allows the user to perform 2D edits as painting or scissoring, which are then propagated to the other images using as guide a voxel-based representation. However, the method has some limitations:

it assumes lambertian surfaces to perform the voxel reconstruction, and it does not support sparse strokes-based edit propagation.

Zhang et al. [ZWGS02] morph between two input light fields, by requiring the user to define equivalent zones in both input light fields. A second approach [COSL05] allows the user to interactively deform an object represented by a light field. Both methods perform deformation by using a warping operator, which is included in Lightshop's operations set [HC07]. Those methods perform deformation in light fields, while our method is designed to interactively change their appearance.

Several techniques for efficient propagation of sparse edits in an image have been recently proposed, spanning a number of different applications. These include image colorization [LLW04] or tone adjustment [LFUS06]. Also, sparse edits propagation methods have been generalized to any application where parameters should be propagated [PL07, AP08, XLJ*09, XYf10]. Those methods shared a common approach, defining similarity metrics between pixels, can be further accelerated by performing the propagation with a downsampled version of the data and then upsampling the solution [KCLU07].

Finally, other authors have introduced structured data editing which is also related to our problem. Bidirectional Texture Functions (BTFs) [DvGNK99] is another structured image-based data used to represent real-world captured appearance. From all the works appeared aiming to edit this data, the approach of Xu et al. [XWT*09] propagates sparse edits in the full data set, as our method does. However, they use additional information present in the BTFs to formulate the affinity metric, and they have to use an out-of-core propagation implementation when editing large BTFs. In contrast, our method works only with appearance and spatial information, and it's designed to perform edit propagation in core, as it's based on smartly downsampling the light field.

3. Edit propagation

Coarse stroke-based edit propagation [LFUS06, PL07, AP08, XLJ*09] allows fast editing in images by asking the user to input just a few sparse strokes, and propagating them to the rest of the image. This propagation is usually performed keeping in mind two principles: first, pixels covered by one stroke should keep the appearance given by the user as much as possible. And second, near pixels with similar appearance should receive similar edits. To account for these two principles, it is necessary to define a mathematical formulation of the propagation, where the final propagated edits in a pixel depends on the explicit edit performed over that pixel, and also in the edits performed over pixels with similar appearance.

The recent work by An and Pellacini, AppProp [AP08], allows an efficient editing of complex spatially-varying datasets, while being able to propagate sparse edits over discontinuous regions. It would in principle be a good candidate for light field editing as well, but as we will see, it presents several hard limitations. The key of the AppProp approach is the imposition that close-by pixels with similar appearance have similar edits. This allows imprecise user strokes to be correctly propagated over the full data. Initially, the user does some strokes g_i with strength $w_i \in [0,1]$ in some parts of the image, and these initial edits are propagated to the rest of the image, yielding the final edits that will be performed, e_i. These final edits e_i are obtained by minimizing the following energy function:

$$\sum_{i,j} w_j z_{ij} (e_i - g_j)^2 + \lambda \sum_{i,j} z_{ij} (e_i - e_j)^2 \qquad (1)$$

where z_{ij} is the affinity between samples i and j, and λ is the variable that controls the relative contribution between both terms of the summation. The first term imposes the constraint specified by the user strokes, while the second one ensures that pixels with high affinity are similarly edited.

The affinity metric z_{ij} between two pixels i, j models their similarity considering both appearance and the spatial distance between them. It is defined as:

$$z_{ij} = \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{\sigma_a}\right) \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_s}\right) \qquad (2)$$

where $\mathbf{c}_i$ and $\mathbf{x}_i$ are appearance and spatial location vectors of pixel i, and σ_a and σ_s are the propagation controllers that model how much each feature affects the propagation.

Given that the energy function from Equation 1 is quadratic, it can be minimized solving a linear system with n variables (the affinity matrix is of size $n \times n$, where n is the size of the data). The system is efficiently solved by using an stochastic column sampling approach, in which the affinity matrix, being close to low rank, is approximated by m linearly independent columns ($m << n$). This leads to a time and space complexity of $O(m^2 n)$ and $O(mn)$ respectively, where m is the number of columns randomly sampled. So even though the linear system can be solved very efficiently, its complexity is still linear with the data size. This is an important limitation when working with large elements, such as 4D light fields, which renders the method impractical: not only can user edits take a very long time to propagate, but the data may not even fit in memory. Our technique, which we describe in the following section, is designed to overcome these problems.

4. Overview of our method

As we have seen, AppProp [AP08] allows the user to perform complex edits by automatically propagating sparse and imprecise user strokes. Although it yields impressive results in images, we deal in this work with light fields, more complex both in terms of structure and size. Light fields are enormous quantities of data which capture all the light rays (up to a sampling rate) within a scene. To do this, a ray is typically represented by the intersection of it with two planes, namely the plane of the camera (uv plane) and the focal plane (st plane) [LH96]. These characteristics (the large data size and the four-dimensional parametrization) are what makes the use of AppProp *as is* impractical for editing light fields. We outline here the two main problems, together with the proposed solution.

First, the propagation and the affinity metric are defined to work on a two-dimensional spatial domain, while the light field space is defined by four dimensions. Therefore, it is necessary to define a new affinity metric adapted to the four-dimensional spatial domain of the light field.

The *second* consideration to take into account is related

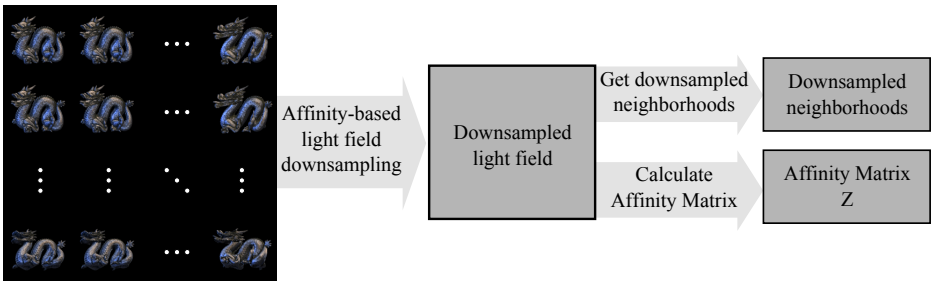

Figure 1: *An overview of our proposed preprocessing. From the initial light field data, we get the downsampled elements. Those downsampled elements are then used to precalculate the affinity matrix Z, that later will be used in the propagation process. Also, the neighborhood of each downsampled element is got, to allow fast upsampling when editing.*

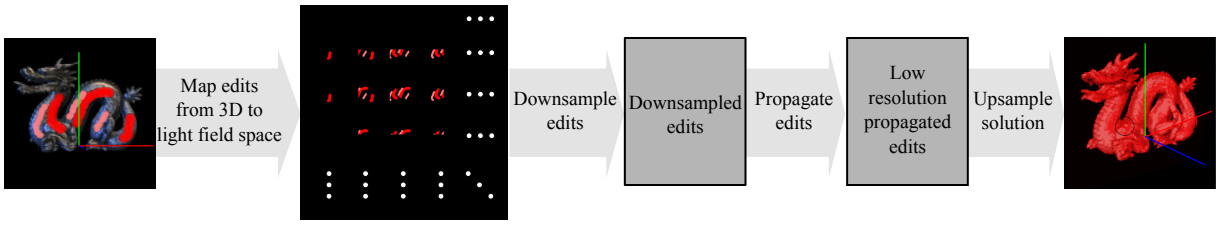

Figure 2: *Our interactive editing pipeline: Over the 3D representation of the light field, the user can draw sparse strokes. To propagate them, edits have to be first mapped from screen space to the light field, and then downsampled. Once downsampled, they are propagated in the downsampled domain. Finally, the solution is upsampled.*

with the efficiency of the method, whose complexity is linear with the size of the data. To overcome this, we devise the working strategy of performing the propagation in a downsampled version of the data and then upsampling the edits to the full resolution data. However, the downsampling process is not straightforward, because the edit propagation process is guided by an affinity function which usually accounts both for appearance similarity and spatial proximity. As a consequence, a straightforward downsampling in the spatial 2D domain would be problematic, since pixels close in 2D space could be clustered together in the low resolution version despite possibly having great dissimilarity; and subsequently pixels which should not receive similar edits would in fact do. We therefore downsample not only in the spatial domain, but also accounting for other features, ensuring that similar (in a multi-dimensional way) pixels will receive similar edits. This implies downsampling the data considering all the dimensions of the affinity metric used when propagating edits. To do this, we follow an approach similar to the work of Xu et al. [XLJ*09], mapping all pixels to an affinity space defined by the dimensions of the affinity metric and then performing the downsampling in that space. Upsampling the propagated edits back to full resolution is not trivial either, as the next section explains.

Based on our new metric and downsampling-upsampling strategy, we have developed a framework to efficiently edit light fields, providing the user rapid feedback, and which can propagate sparse strokes done by the user over the full light field. We divide our pipeline in two steps, explained below.

Preprocessing step. During this step (depicted in Figure 1), we downsample the light field, and then create the affinity matrix with the downsampled data. Additionally, we precalculate the neighborhood of each downsampled pixel, since it will be used during the editing process.

Interactive editing step. The user first draws some sparse edits over a 3D representation of the light field. Once the edits are mapped from screen space to the four dimensional light field space, they are downsampled. Those downsampled edits feed the propagation process, returning a downsampled version of the solution. Finally, the low resolution propagated edits are upsampled and applied over the full light field. Figure 2 shows the complete pipeline of this step.

5. Efficient light field edit propagation

Having explained the method as a whole in the previous section, we offer here a detailed explanation of those steps of the pipeline which were specifically devised to adapt An and Pellacini's approach to work with light fields and thus are significantly different from the AppProp scheme.

5.1. Definition of a new affinity metric

The affinity metric defined by Equation 2 could be used to guide the propagation in light fields by considering the light field as a large 2D image formed by all the different tiles or slabs. However, using this metric would lead to incorrect propagation because the distance between points (second term in Equation 2) can not be correctly calculated for a light field if it is parametrized in 2D. This is illustrated in Figure 3. For two points which are neighbors in a light field, using 2D spatial coordinates yields a distance between them equal to the size of a tile (*sizetile*), whereas for a 4D parametrization the distance between them is indeed 1 (in arbitrary distance units). Thus, instead of using the spatial term from Equation 2, we substitute it with two new terms to model the full 4D light field space (i.e. *st* and *uv* planes), which leads to our new affinity metric:

$$z_{ij} = \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{\sigma_a} - \frac{\|\mathbf{st}_i - \mathbf{st}_j\|^2}{\sigma_{st}} - \frac{\|\mathbf{uv}_i - \mathbf{uv}_j\|^2}{\sigma_{uv}}\right) \quad (3)$$

where $\mathbf{c}_i = (r_i, g_i, b_i)$ are the appearance values of pixel i. Instead of, or as well as, RGB tuples, other measures of appearance could be used. For the present implementation we have chosen RGB values because they perform well and are the simplest to obtain. Vectors $\mathbf{st}_i = (s_i, t_i)$ and $\mathbf{uv}_i = (u_i, v_i)$ contain the spatial coordinates in the 4D light field space. Finally, σ_a, σ_{st} and σ_{uv} control how much each feature (appearance and spatial distance in each light field plane) affects the measure of similarity, and subsequently, the propagation of the edits. It is important to note that all features are normalized to the resolution of each dimension.

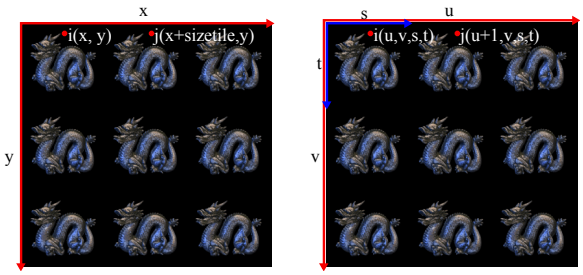

Figure 3: *Using 2D (*left*) vs. 4D (*right*) coordinates when building the affinity matrix. If 2D coordinates were used, the spatial distance between points i and j would be* sizetile*, leading to incorrect propagation. Using 4D coordinates the distance equals* 1*, which is correct since both points are neighbors in the light field space.*

5.2. Building the affinity space

The affinity metric defined in Equation 3 can be expressed as:

$$z_{ij} = \exp(-\|\mathbf{f}_i - \mathbf{f}_j\|^2) \tag{4}$$

where $\mathbf{f}_i = (\frac{\mathbf{c}_i}{\sqrt{\sigma_a}}, \frac{\mathbf{st}_i}{\sqrt{\sigma_{st}}}, \frac{\mathbf{uv}_i}{\sqrt{\sigma_{uv}}})$ is the seven-dimensional vector that stores the appearance values and the 4D light field coordinates of pixel i; all of them scaled by the reciprocal of the square root of their associated σ values. Those features can be understood as the coordinates of the pixel in a seven-dimensional space, where each dimension d is in the range $[0, {\sigma_d}^{-1}]$.

Following Xu et al. [XLJ*09], we call our seven-dimensional space the *affinity space* and, since Equation 3 decreases with the squared difference between pixels' features, it is trivial to note that the closer the pixels are in affinity space, the bigger their similarity is.

5.3. Downsampling the light field

Downsampling data may be understood as a type of clustering of the elements from the original data, where the elements inside a cluster C are represented by one representative element, which we will call $\tilde{j}$. The ideal downsampled light field would minimize the differences between each pixel i in the original light field and the representative of the cluster, $\tilde{j}$. In our context, minimizing the differences is equivalent to having the highest affinity.

Following the observation that pixels close in affinity space have big similarity, minimizing the difference between pixel i and its representative $\tilde{j}$ is performed by clustering groups of pixels in our seven-dimensional affinity space, and then getting a single representative for the cluster.

Thus, to find the downsampled clusters in the light field, we first map all pixels into affinity space. Then, we downsample the elements in a top-down approach, by recursively subdividing the space into clusters. It is actually very similar to how a BVH (bounding volume hierarchy) is constructed, but without keeping the hierarchical structure, as we are just interested in the final clusters. The algorithm stops when it reaches a cluster with the defined number of pixels. This top-down approach has been chosen because of its simplicity and speed, and also because it guarantees that all clusters represent the same number of pixels. Once the cluster is obtained, the representative is calculated by averaging the features of the elements in the cluster.

5.4. Downsampling the user strokes

Despite propagating the edits on downsampled data, user strokes are done over the full-resolution light field. It is then mandatory to downsample those strokes, in order to get a suitable input to perform the propagation in the low resolution data.

From the original strokes g_i done by the user and their associated strengths w_i (following the notation explained in Section 3), we calculate the downsampled $\tilde{w}_{\tilde{j}}$ and $\tilde{g}_{\tilde{j}}$ as:

$$\tilde{w}_{\tilde{j}} = \frac{1}{k_p} \sum_{i \in C} w_i * z_{i\tilde{j}} \tag{5}$$

$$\tilde{g}_{\tilde{j}} = \frac{1}{k_p} \sum_{i \in C} g_i * z_{i\tilde{j}} \tag{6}$$

where C is a cluster of pixels in the original light field, $\tilde{j}$ is the representative of that cluster, and $z_{i\tilde{j}}$ the similarity between $\tilde{j}$ and each pixel i in C. $k_p = \sum(z_{i\tilde{j}})$ is introduced to ensure energy conservation.

5.5. Upsampling the propagated edits

Once edit propagation has finished, a low resolution result $\tilde{E}$ is obtained. Then, a full resolution propagated E version has to be obtained from the downsampled $\tilde{E}$. A very simple approach to get the upsampled value in pixel i would be just $E_i = \tilde{E}_{\tilde{j}}$, being $\tilde{j}$ the downsampled representative of i.

This approach, however, does not take into account the fact that one pixel assigned to one cluster can also have big similarity with other cluster representatives. To solve this problem, we take advantage of the available full resolution data, which can be used to guide the upsampling process, similar to Joint Bilateral Upsampling [KCLU07]. When calculating the upsampled solution E_i in pixel i, we account not only for the contribution of its representative $\tilde{j}$, but also for the contribution of the neighborhood of $\tilde{j}$, weighted by its similarity with i. This similarity can be calculated because we still have the original high resolution light field. So, the upsampled solution E is calculated as:

$$E_i = \frac{1}{k_p} \sum_{\tilde{n} \in neig(\tilde{j})} \tilde{E}_{\tilde{n}} * z_{i\tilde{n}} \tag{7}$$

where $z_{i\tilde{n}}$ is the similarity between pixel i and $\tilde{n}$, and k_p is the normalizing term, $k_p = \sum(z_{i\tilde{n}})$.

To obtain the neighborhood of each representative $\tilde{j}$, as they are no longer in a uniform grid, we cannot query an array to perform the search. Instead, we search the k nearest neighbors in a kd-tree, imposing also a maximum radius defined by a minimum affinity threshold. For efficiency reasons, this search is done in preprocess step, storing, for each representative $\tilde{j}$, the indices of its neighbors.

6. Results

We have tested our framework for editing light fields on the Old Stanford Light Fields Archive [Sta], where we have performed some color edits interactively. Color edit propagations have been chosen because they show how multiple parameters edits can be easily performed at the same time, although any other edits based in propagating parameters would be also possible (e.g. exposure, contrast, saturation, color temperature). Two examples of the edits done are shown in Figure 6, where the dragon and the buddha light fields have been edited. The propagation of those edits, performed on a PC with an Intel Core2 P8700 (2.53GHz, 2 cores) and 2GB memory, took roughly 0.50 seconds, with the resolution of both light fields being $(32 \cdot 256)^2$ pixels, and downsampled by a factor of 32^2.

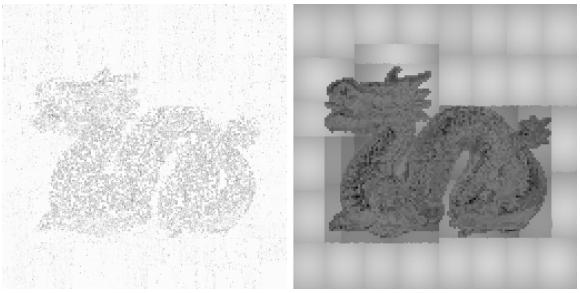

Figure 4: *Similarity between pixels and their downsampled representatives when downsampling by a rate of* 2^2 *(*left*) and* 64^2 *(*right*). The whiter the image is, the more affine with their downsampled versions pixels are.*

As the downsampling rate augments, propagation times logically decrease, since the number of elements to be handled by the algorithm is reduced. This reduction is linear with the downsampling rate. However, the higher the downsampling rate is, the greater the error produced by the approximation. This is because when the size of a cluster augments, due to higher downsampling rates, the pixels inside the cluster are further in affinity space, and thus their similarity with the representative is lower. An example of this is shown in Figure 4, which depicts, for two different downsampling ratios, the similarity between each pixel and its representative (the whiter, the more similar). With a low downsampling ratio (2^2 for the left image) the similarity is quite high, while the right image (downsampling ratio 64^2) exhibits much darker areas, indicating significative differences between each pixel and its assigned representative.

To validate that high differences between downsampled representatives and high resolution images do not affect the result from a purely visual perspective, we compare the edits between a downsampled propagated image and an original one. To perform the comparison, we use a reduced version of a light field with size $(2 \cdot 256)^2$, in order to have the high resolution propagation process fit in memory. The results even with a high downsampling rate are, as shown in Figure 5 visually correct, although some minor errors appear.

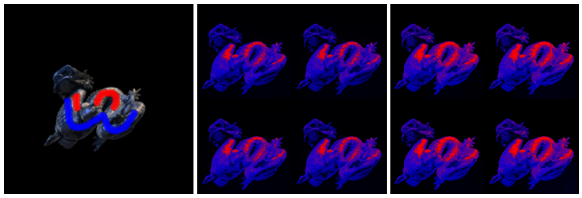

Figure 5: Left: *3D view of a light field with the sparse edits performed by the user.* Middle: *Result of propagating the edits on the original (without downsampling) light field of size* $(2 \cdot 256)^2$. Right: *Result when the edits are propagated on a downsampled version of the light field, the downsampling ratio being* 32^2.

The method has, however, some limitations. The most important one is that the complexity of the method is still linear with the light field size: time complexity is $O(m^2 n/ratio)$ and space complexity $O(mn/ratio)$, n being the data size, m the randomly sampled columns for the low-rank approximation, and $ratio$ the downsampling ratio. Thus, with very large light fields, the time will still grow linearly, unless the downsampling ratio is increased. Also, in terms of memory requirements, in addition to the clusters storage it is necessary to keep the correspondence between pixels with their clusters.

7. Conclusions and future work

We have proposed an efficient method for editing light fields, based on a sparse edit propagation scheme. Our framework allows the user to perform coarse sparse editions over a 3D representation of the light field, and then propagate them over the full-resolution data, guiding the propagation by the similarity between the pixels. Our method allows efficient calculations, even in very large light fields, by downsampling them in our affinity space, in a way that pixels with high affinity are downsampled together. Propagation is then performed over the downsampled data, and the result obtained is upsampled to yield the final solution.

Currently our method performs downsampling by recursively subdividing the pixels in clusters. In the future we would like to explore other techniques to downsample the data. Adding new edit operations to our framework would be also interesting, since the current implementation only supports color modifications. Nevertheless, we believe our approach can become a useful tool in the somewhat unexplored area of light field editing, where the sheer size of the data renders existing edit propagation methods impractical.

8. Acknowledgements

We would like to thank the reviewers for their insightful comments. Many thanks also to the Stanford University Computer Graphics Laboratory for making their Light Field Archives publicly available. This research has been

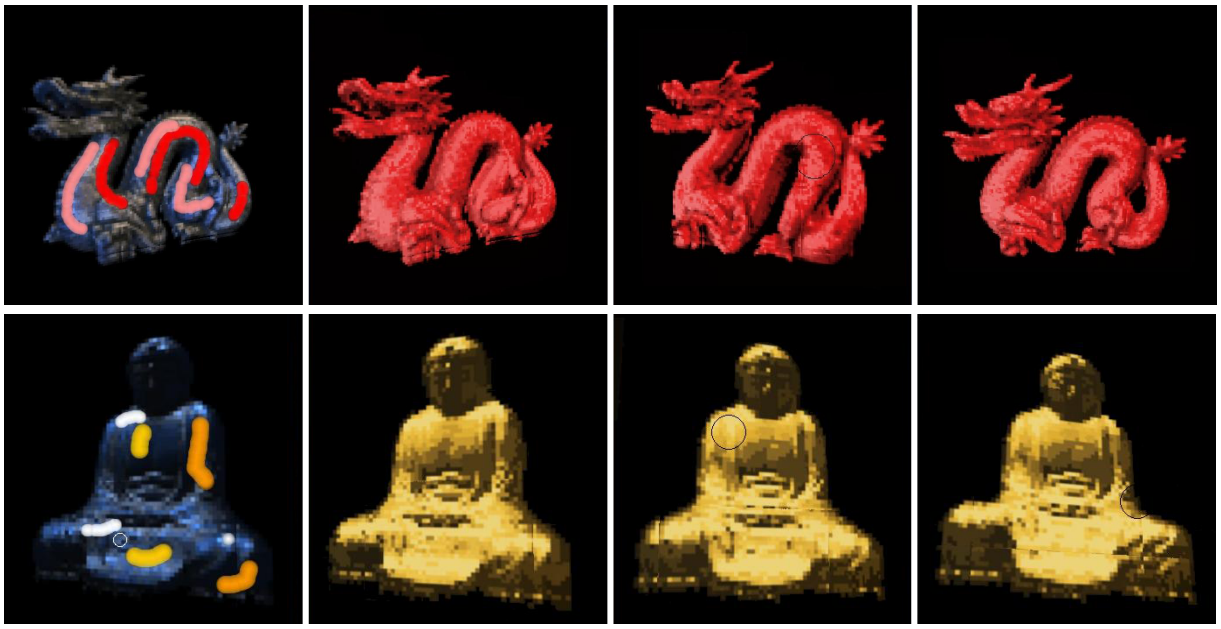

Figure 6: *Strokes drawn over the 3D view of the light field (*left*), and the resulting propagated edits in different views for the dragon (*top row*) and buddha light fields (*bottom row*). Please note that the low resolution of the buddha light field is inherent to the original light field and not related to the downsampling-upsampling process.*

funded by a Marie Curie grant from the Seventh Framework Programme (grant agreement no.: 251415), the Spanish Ministry of Science and Technology (TIN2010-21543) and the Gobierno de Aragón (projects OTRI 2009/0411 and CTPP05/09). Belen Masia is supported by a FPU grant from the Spanish Ministry of Education.

References

[AP08] AN X., PELLACINI F.: Appprop: all-pairs appearance-space edit propagation. *ACM Trans. Graph. 27* (August 2008), 40:1–40:9. 1, 2

[CNR08] COSSAIRT O., NAYAR S., RAMAMOORTHI R.: Light field transfer: global illumination between real and synthetic objects. In *ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 57:1–57:6. 1

[COSL05] CHEN B., OFEK E., SHUM H.-Y., LEVOY M.: Interactive deformation of light fields. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 139–146. 2

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18* (January 1999), 1–34. 2

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *Proc. of the 23rd annual conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 43–54. 1

[HC07] HORN D. R., CHEN B.: Lightshop: interactive light field manipulation and rendering. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 121–128. 1, 2

[KCLU07] KOPF J., COHEN M. F., LISCHINSKI D., UYTTENDAELE M.: Joint bilateral upsampling. *ACM Trans. Graph. 26* (July 2007). 2, 4

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R.: Interactive local adjustment of tonal values. *ACM Trans. Graph. 25* (July 2006), 646–653. 2

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 31–42. 1, 2

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Trans. Graph. 23* (August 2004), 689–694. 2

[Ng05] NG R.: Fourier slice photography. *ACM Trans. Graph. 24* (July 2005), 735–744. 1

[PL07] PELLACINI F., LAWRENCE J.: Appwand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph. 26* (July 2007). 2

[SK02] SEITZ S., KUTULAKOS K.: Plenoptic image editing. *Intnl. Journal of Computer Vision 48*, 2 (2002), 115–129. 1

[Sta] The (Old) Stanford Light Fields Archive. http://www-graphics.stanford.edu/software/lightpack/lifs.html. [Online, accessed 22-March-2011]. 5

[VRA*07] VEERARAGHAVAN A., RASKAR R., AGRAWAL A., MOHAN A., TUMBLIN J.: Dappled photography: mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Trans. Graph. 26* (July 2007). 1

[XLJ*09] XU K., LI Y., JU T., HU S.-M., LIU T.-Q.: Efficient affinity-based edit propagation using k-d tree. *ACM Trans. Graph. 28* (December 2009), 118:1–118:6. 2, 3, 4

[XWT*09] XU K., WANG J., TONG X., HU S.-M., GUO B.: Edit propagation on bidirectional texture functions. *Computer Graphics Forum 28*, 7 (2009), 1871–1877. 2

[XYf10] XIAO C., YONGWEI N., FENG T.: Efficient edit propagation using hierarchical data structure. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints (2010). 2

[ZWGS02] ZHANG Z., WANG L., GUO B., SHUM H.-Y.: Feature-based light field morphing. *ACM Trans. Graph. 21* (July 2002), 457–464. 2