Directional Super-Resolution by means of Coded Sampling and Guided Upsampling

David C. Schedl Clemens Birklbauer Oliver Bimber
Johannes Kepler University Linz
forename.surename@jku.at

Abstract

We present a simple guided super-resolution technique for increasing directional resolution without reliance on depth estimation or image correspondences. Rather, it searches for best-matching multidimensional (4D or 3D) patches within the entire captured data set to compose new directional images that are consistent in both the spatial and the directional domains. We describe algorithms for guided upsampling, iterative guided upsampling, and sampling code estimation. Our experimental results reveal that the outcomes of existing light-field camera arrays and light-stage systems can be improved without additional hardware requirements or recording effort simply by realignment of cameras or light sources to change their sampling patterns.

1. Introduction and Contributions

Sampling and processing images under directionally varying (viewing or lighting) conditions is the key to several modern visual effects production techniques in digital photography and movie making. Image-based rendering of light fields [11, 9] or spatio-temporal recordings [18] and image-based relighting of light-stage data [4] generally do not rely on depth estimation or image correspondences, as this can be difficult or even impossible to compute for realistically complex sceneries. To prevent undersampling artifacts, however, a substantial number of images from many directions must be captured and combined. This leads to complex and dense camera- or light-arrays, and to performance constraints that are due to bandwidth limitations. In this paper, we present a simple guided super-resolution technique for the directional domain that does not require scene depth, disparity estimation, or exact image correspondences but searches for best-matching multidimensional (4D or 3D) patches within the entire captured data set to compose new directional images that are consistent in both the spatial and the directional domains. The applicability of our approach is only limited by the maximum disparity/distance among neighboring sample positions. We present results for existing light-field camera arrays and light-stage systems, and show how directional aliasing artifacts of these devices can be improved simply by realigning their sampling positions.

2. Related Work

2.1. Spatial and Temporal Super-Resolution

A rich literature on super-resolution techniques exists. **Single-image super-resolution** techniques for example, use patch databases from various scales of the input image (e.g. [8, 5]). For **temporal super-resolution**, these methods were extended to the time domain by utilizing the recurrence of space-time patches across spatio-temporal scales of the input video to reduce motion aliasing and motion blur (e.g. [17]). One drawback of most light-field cameras is their low spatial resolution. Therefore, various **light-field super-resolution** approaches, for computing single high-resolution images, have been proposed [1, 7]. A hybrid imaging system that consists of a regular high-resolution camera and a low-resolution light-field camera was presented in [2]. The input of the high-resolution camera serves as guidance for upsampling the spatial resolution of the light field. Recent compressive sensing approaches capture light fields by placing masks in front of an image sensor of a regular camera (e.g. [12, 14]), but require computationally expensive reconstruction and only allow small perspective changes.

2.2. Directional Super-Resolution

Directional undersampling in the case of too few perspective recordings or too large disparities can also lead to strong visual artifacts (see [3] for an analysis). **View interpolation** reduces these artifacts by upsampling the number of views used for rendering, for example via an estimation framework that utilizes the epipolar plane structure of a light field [20] or trained Gaussian mixture models for light-

field patches [15], where disparity values are estimated for each patch.

For **image-based relighting** a limited number of lighting bases introduces aliasing artifacts for high-frequency lighting effects (e.g., sharp shadows, highly specular surface reflections, and for more complex light transports). The limits are usually hardware constraints, such as the frame rate of the cameras or activation time of LEDs [4]. Therefore, one way of avoiding artifacts is to upsample the recorded set of lighting bases (e.g. [13]). In [19], the number of images for relighting was drastically reduced by using spherical harmonics. However, due to the small set of local lighting bases, aliasing artifacts continued to be introduced for high-frequency lighting cases. In [6] the lighting bases are upsampled by segmenting the scene into layers (highlights, diffuse, shadows) and by handling them separately. Flow algorithms are applied to the highlights and shadows layer (the layers that cause the strongest interpolation artifacts). Incorrect segmentation or wrong flow correspondences introduce errors.

Finding exact image correspondences for depth reconstruction, disparity estimation, or optical flow computations, however, is only possible for adequately textured isotropic sceneries. More realistic scenes with, for instance, anisotropically reflecting surfaces or transparent refracting objects do not support robust estimation of image correspondences. Nevertheless, these are the cases that require image-based rendering rather than geometry-based rendering. If dense depth estimation and optical flow computation are possible, then image-based light-field rendering and relighting are superfluous. Linear view synthesis [10], that calculates novel views from focal stacks without requiring image correspondences, covers only a 3D subset of 4D light fields. In this paper, we present an entirely image-based and simple guided super-resolution technique for increasing directional resolution without reliance on image correspondences as required for depth-based morphing or interpolation based on optical flow.

3. Upsampling Camera Arrays

Let us assume an input grid $D_i(S,T,U,V)$ of $U \times V$ directional samples of images with spatial resolution $S \times T$ (e.g., as recorded with a $U \times V$ array of cameras, each having a resolution of $S \times T$ pixels). Our goal is to upsample the directional resolution to an output grid D_o by an upsampling factor of f while retaining the spatial resolution at each sample. For guided upsampling, we generally consider n directional guidance regions $G_{r=0..n-1}$ (areas with denser sampling) that are distributed within the sampling area. The sampling rate of each G_r equals the sampling rate of D_o, and is by a factor f higher than the sampling rate of D_i. As the name suggests, these regions of higher sampling rate serve as guidance for our directional upsampling. We extract high-frequency information from these guidance areas and add these details to novel views. How many guidance regions are used, how large they are, and where they are located for data acquisition is discussed in section 3.2.

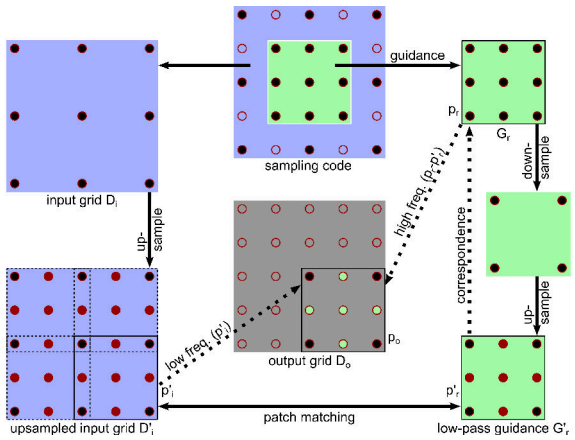

Figure 1. **Guided upsampling.** Black dots in the sampling code represent captured samples, while empty circles are samples computed by our method. The guidance area is indicated in green. Red dots are interpolated samples. A processed patch is indicated by the black rectangle. Green filled circles illustrate upsampled samples with high frequency details from the guidance patch p_r and p'_r, respectively.

Our first step is to low-pass filter each G_r in the directional domain to simulate the sampling rate of D_i. This is achieved by downsampling (with nearest neighbor sampling) all G_r by the factor $1/f$, and subsequently upsampling the results by f again while linearly interpolating the missing directional samples (i.e., the corresponding images at each directional sample position). Note that up- and downsampling are applied exclusively to the directional domain. The spatial domain (i.e., the resolution of each sample image) remains unaffected. This results in (directionally) low-pass filtered guidance regions G'_r with partly interpolated image samples. The next step is to upsample D_i by f while linear interpolation is again applied to approximate the missing directional samples. This also results in an upsampled input grid D'_i with partly interpolated image samples. Note that, again, upsamling is applied only to the directional domain. Note also that D'_i, D_o, G'_r, and G_r now all have the same sampling rate. We synthesize D_o as follows: For a given s,t,u,v coordinate in D_o, we extract a four-dimensional neighborhood patch p'_i of size $\delta_s \times \delta_t \times \delta_u \times \delta_v$ around $D'_i(s,t,u,v)$, and find the closest match p'_r in all G'_r. The final output patch is then approximated with $p_o = p'_i + (p_r - p'_r)$, where p_r is the patch in G_r that corresponds to p'_r in G'_r. Note that the subtraction of p_r and p'_r corresponds to high-pass filtering. Thus, we add only high frequencies $(p_r - p'_r)$ to the low frequencies

Algorithm 1 Guided upsampling

function GUIDED_UPSAMPLING($D_i, G, n, \delta_s, \delta_t$)
  D_o=init()
  f=samplingrate(D_i)/samplingrate(G_0)
  (δ_u, δ_v)=size(G_0)
  for $r = 0..(n-1)$ **do**
    G'_r=downupsample(G_r, f)
  end for
  D'_i=upsample(D_i, f)
  for all s, t, u, v in D_o **do**
    if u, v valid /*same sampling pattern*/ **then**
      p'_i=getpatch($D'_i, s, t, u, v, \delta_s, \delta_t, \delta_u, \delta_v$)
      p'_r=patchmatch(G', p'_i)
      p_r=getcorrespondingpatch(G, p'_r)
      $p_o = p'_i + (p_r - p'_r)$
      averrageandsetpatch(D_o, s, t, u, v, p_o)
    end if
    if u, v not interpolated in D'_i **then**
      $D_o(u, v) = D'_i(u, v)$
    end if
  end for
  return D_o
end function

contained in p'_i for estimating the output patch. Note that multiple estimates in D_o that result from patch overlaps are averaged. Finally, we transfer all non-interpolated sample images of D'_i to D_o. Thus, our estimations are overwritten whenever a physically captured sample is available.

Figure 1 illustrates our method based on the example of a regular 3×3 input grid that is upsampled to a regular 5×5 output grid with $n = 1$ guidance region of an $f = \frac{5}{3}$ higher sampling rate and 3×3 guidance samples. In this case, δ_u and δ_v are 3 (the size of the guidance region). Note that p'_i and p'_r are only comparable if they share the same sampling pattern. Since p'_r consists of non-interpolated samples at the four corners and interpolated samples elsewhere, we must constrain u, v during patch matching to coordinates that lead to patches p'_i with the same sampling pattern (referred to as *cells* in the following). For our example in figure 1, this applies only to four u, v coordinates (dashed in D'_i). Other u, v coordinates are invalid and cannot be considered. Algorithm 1 summarizes our guided upsampling approach.

3.1. Iterative Guided Upsampling

The sampling pattern of the guidance patches and the upsampled patches must be identical to be comparable (sampling rate and patch size). This implies that the guidance regions must be densely sampled for large upsampling factors f. To avoid that an unnecessary large number of guidance samples must be recorded, we support upsampling over multiple iterations. Our iterative upsampling requires the sampling code to contain nested guidance regions with multiple sampling rates. Figure 2 illustrates an example of a sampling code with two guidance regions of $f_2 = \frac{5}{3}$ (green/blue) and $f_1 = \frac{9}{5}$ (red/green). In the first iteration, the input grid is upsampled with the lowest upsampling factor. In our example, the f_2-guidance is used to upsample

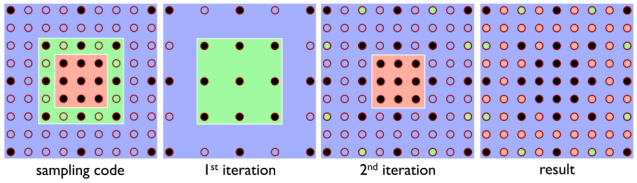

Figure 2. **Iterative guided upsampling.** Two embedded guidance regions consisting of different sampling rates (green and red). Black dots in the sampling code represent captured samples, while empty circles are samples computed by upsampling in two iterations. The color of filled circles indicate which sample is computed in what upsampling iteration.

the 3×3 input grid to a 5×5 output grid. This is identical to the example shown in figure 1. In the second iteration, the 5×5 output grid of the first iteration is used as input grid together with the f_1-guidance to be upsampled to a 9×9 output grid. Our iterative guided upsampling is summarized in algorithm 2, where m is the number of iterations, and $G = \{G_{0..n-1}^{0..m-1}\}$ are the n directional guidance regions per iteration (i.e., $n = 1$ and $m = 2$ with G_0^0 and G_0^1 for the example shown in figure 2).

3.2. Estimation of Sampling Code

Our upsampling algorithm can only be applied to data that has been sampled with an adequate sampling code. Therefore, the estimation of valid sampling patterns is essential. Let us assume a given upper limit of the number of samples we want to capture $c_{\max}$. This is, for example, the largest number of cameras available for a light-field camera array. Furthermore, we want to assume a given output resolution u_o, v_o (i.e., the number of final samples we want to generate). The goal is to find a sampling code which supports upsampling the data captured with it to the desired output resolution without the code's number of samples exceeding the defined upper limit. Code estimation generally corresponds to modeling the guidance regions: their number, their size, their positions within the sampling grid and their nesting levels that correlate to the number of upsampling iterations. Here, we apply the following basic guidelines:

1) A larger number of iterations reduces the number of samples required, but too many iterations will lead to in-

Algorithm 2 Iterative guided upsampling

function IT_GUIDED_UPSAMPLING($D_i, G, n, m, \delta_s, \delta_t$)
  D_o=init()
  for q=$(m-1)..0$ **do**
    D_q=GUIDED_UPSAMPLING($D_i, G^q, n, \delta_s, \delta_t$)
    D_o=merge(D_o, D_q)
    D_i=D_o
  end for
  return D_o
end function

Algorithm 3 Sampling code estimation

```
function GENERATE_SAMPLING_CODE(u_o, v_o, c_max)
    D_0=SC=init(u_o, v_o)
    f=determineFactors(D_0)
    m_max=size(f) /*upper limit of iterations*/
    for m=1..m_max do /*# of tested iterations*/
        D_m=downsample(D_{m-1}, f_m)
        n_max=numberOfCells(D_m)
        for n=n_max..1 do /*# of tested guidance areas*/
            for j=1..m do /*for each iteration*/
                G^{j-1}=posguidance(n,cell(D_j))
            end for
            SC=merge(G, D_m)
            if size(SC) ≤ c_max then
                return SC
            end if
        end for
    end for
end function
```

accurate patch matching and upsampling artifacts in the final results. In our experiments, more than two iterations were not considered to be useful. 2) High-resolution samples should be captured such that they are representative of the entire sampling range. Therefore, a maximum number of guidance regions should be evenly distributed within the sampling grid. 3) The distance between patches to be upsampled and the nearest guidance regions should be as small as possible. Otherwise representative matches will not be found. 4) As explained earlier, the sampling pattern of guidance patches must match the sampling pattern of patches to be upsampled. If multiple sampling patterns are required, they should also be evenly distributed.

Based on guidelines 1 and 2, our goal is to estimate a sampling code with a maximum number of guidance areas and a minimum number of iterations—all under the constraint of not exceeding $c_{\max}$ samples. Algorithm 3 explains our estimation approach. From the uniform output grid of resolution u_o, v_o, we first determine the maximum number of iterations $m_{\max}$ and the corresponding sequence of directional upsampling factors $f_{1..m_{\max}}$. These factors (varying over all iterations) are all valid sampling rates that support consecutive downsampling of a grid of the output resolution u_o, v_o=size(D_0). For the camera array examples discussed in section 3.1, we sought to estimate an optimal sampling code for a 9×9 output resolution of a light field with no more than 25 cameras. For the 9×9 output grid, the maximum number of iterations that can be applied is $m_{\max} = 3$, and the upsampling factors are, $f_1 = \frac{9}{5}$, $f_2 = \frac{5}{3}$, and $f_3 = \frac{3}{2}$. For downsampling a regular grid of rectangular cells, we consecutively transfer every $(\delta_u - 1)^{\text{th}}$ and $(\delta_v - 1)^{\text{th}}$ sample (starting with the 1st) from grid D_{m-1} to grid D_m. Since for our example $\delta_u \times \delta_v$ is 3×3, every second sample is transferred (in both dimensions), resulting in grids D_0 of 9×9, D_1 of 5×5, D_2 of 3×3, and D_3 of 2×2. D_3 cannot be further downsampled. We then test how many iterations should be applied under our given constraints (starting with 1 and increasing to $m_{\max}$) and—for each iteration count—how many guidance regions should be applied (starting with the largest number, which cannot exceed the number of cells $n_{\max}$ in the grid which corresponds to iteration m, and decreasing to 1). For each guidance count and each iteration count, we determine a guidance code G that contains all n guidance areas of all m iterations. The guidance code G^{m-1} at iteration m corresponds to the sampling rate of D_{m-1}. Finally, the guidance code is merged with the corresponding base grid D_m (i.e., the samples that do not directly belong to the guidance), where D_m is downsampled from D_{m-1}. If the size of the resulting sampling code SC exceeds $c_{\max}$, we need to continue our search by successively (1) reducing the number of guidance patches and (2) increasing the number of iterations. Thus, we stop at the sampling code with the lowest number of iterations and the largest number of guidance areas. The best result for our 9×9 output resolution with 25 cameras example is two iterations and one guidance region (as illustrated in figure 2). We position the n guidance areas for each iteration j independently while satisfying guidelines 3 to 4. The sampling code shown in figure 2, for instance, can improve existing 5×5 light-field camera arrays, such as Point Grey's ProFusion25, by realigning their 25 cameras and by applying guided upsampling to 9×9 views. Figure 3 illustrates other examples for computed sample code patterns. With the 128 cameras of the 16×8 Stanford light-field camera array [21], for example, a 31×13 resolution can be achieved (figure 3a). The 21 perspectives of a Time Track linear camera array [18] can be upsampled to 49 perspectives (figure 3b).

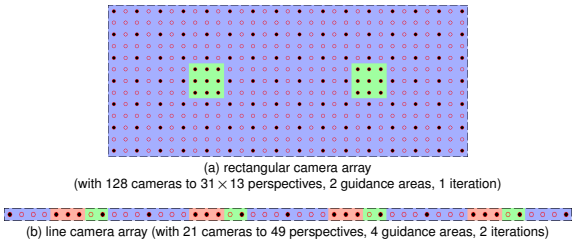

Figure 3. **Different automatically estimated sampling codes.** (a) The 128 cameras of the Stanford camera array upsampled to a light-field resolution of 31×13 perspectives. (b) A linear array of 21 cameras upsampled to 49 cameras.

4. Application to Light Stages

Section 3 illustrates how our method is applied to a regular grid of rectangular cells, as would be required for light-field camera arrays. However, the same technique is applicable to other grid types as long as up- and down-sampling operations to higher and lower topological subdivisions as well as interpolation functions within the corresponding grid cells are clearly defined. This is, for instance, also

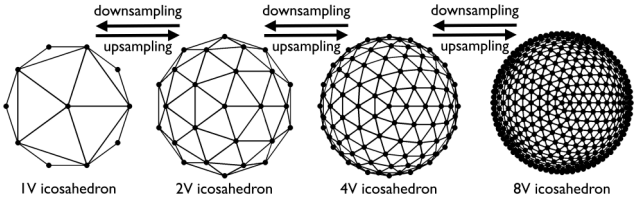

Figure 4. **Icosahedron subdivisions.** 1V with 12 vertices, 2V with 42 vertices, 4V with 162 vertices, and 8V with 642 vertices.

the case for geodesic grids, such as icosahedra, which are the underlying sampling structure for classical light-stage systems. Figure 4 presents 1V, 2V, 4V and 8V subdivisions of an icosahedron that result from successive up- and downsampling. Since the grid cells of icosahedra are triangles rather than rectangles, interpolation required for upsampling is bilinear in the barycentric coordinates of three rather than four basis points. Nevertheless, one essential difference between triangular and rectangular cells exists in terms of patch matching. Whereas triangular cells can have different orientations, rectangular cells always share the same orientation within the sampling grid. As explained earlier, patches can only be compared and matched if they share the same sampling structure, which includes the patches' orientation. Thus, for the icosahedron example, we need to constrain our guided upsampling to patches of equal (or at least very similar) orientations. Figure 5 presents our code for realigning the 362 LEDs of a light stage 6[1] to support upsampling to the 642 light directions of an 8V icosahedron. Here, 80 single-iteration guidance regions of two different sampling structures (two orientations of the triangular grid cells within the same local neighborhood) are distributed. We initially compute the $80/2 = 40$ positions of only one guidance pattern (i.e., one triangle orientation) with centroidal Voronoi tessellation. The remaining guidance patterns (i.e., the other triangle orientation) are evenly distributed within the direct neighborhood of the initial patterns (guideline 4) while choosing those neighbor cells that share a maximum number of sample points. Note that, since icosahedra do not provide symmetric sampling grids, the code patterns for the northern and for the southern hemispheres are different.

[1] We consider an even sampling over a full 6V icosahedron and ignore the flat ground and blocked background regions that are present in physical systems.

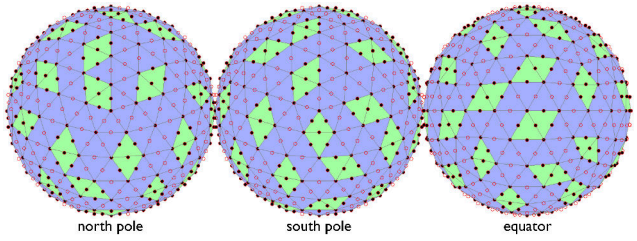

Figure 5. **Estimated sampling code for light stage 6.** The 362 LEDs of light stage 6 (6V icosahedron) are realigned to support upsampling to an 8V icosahedron with 642 LEDs. The renderings show the single-iteration code in different orientations.

5. Implementation

All algorithms were implemented in MATLAB. For patch matching, we apply the kd-tree search with 8 trees and 64 checks per tree of FLANN (Fast Library for Approximate Nearest Neighbors) [16], where 4D patches are represented as 1D feature vectors. Patches are considered similar if the squared difference of their normalized feature vectors is small. Patch matching is carried out exclusively in the luminance channel. Since for each patch a cross-reference to the corresponding color version exists, the approximation of the final patches ($p_o = p'_i + (p_r - p'_r)$, as explained in section 3) is performed in the RGB channels.

6. Results

With the sampling code shown in figure 2, we built an irregular array of 25 cameras. The recorded images were upsampled with our method to a directional resolution of 9×9. For comparison, we realigned the 25 cameras on a regular 5×5 grid and linearly interpolated the captured images to a resolution of 9×9 perspectives. Figure 6 presents refocus renderings of two light-field examples. While the sparse light-field resolution leads to strong bokeh artifacts in out-of-focus regions (due to undersampling), the interpolated light fields are blurred in focussed regions (due to image misalignment caused by the interpolated perspectives). Light fields created with coded sampling and guided upsampling show clear improvements in out-of-focus and in-focus regions. Note that interpolation artifacts are reduced if the synthetic focus approaches the optical focal plane of the camera array. In this case, the disparity among all perspectives is smallest. Figure 7 presents single-perspective images of the *cars* scene computed with our iterative guided upsampling and compares them with the corresponding interpolated images. While minor patch-matching artifacts can be detected in the upsampled results, they are far less noticeable than artifacts caused by interpolation. Note that non-linear interpolation did not improve these results over those of linear interpolation.

Figures 8 and 10 illustrate image-based relighting results that could be achieved if the sampling code shown in figure 5 is used for the light stage 6, rather than a uniform distribution of LEDs. In both cases, 362 LEDs were applied. However, our approach supports effective upsampling to an 8V icosahedron with 642 LEDs—leading to fewer aliasing artifacts for high-frequency lighting effects

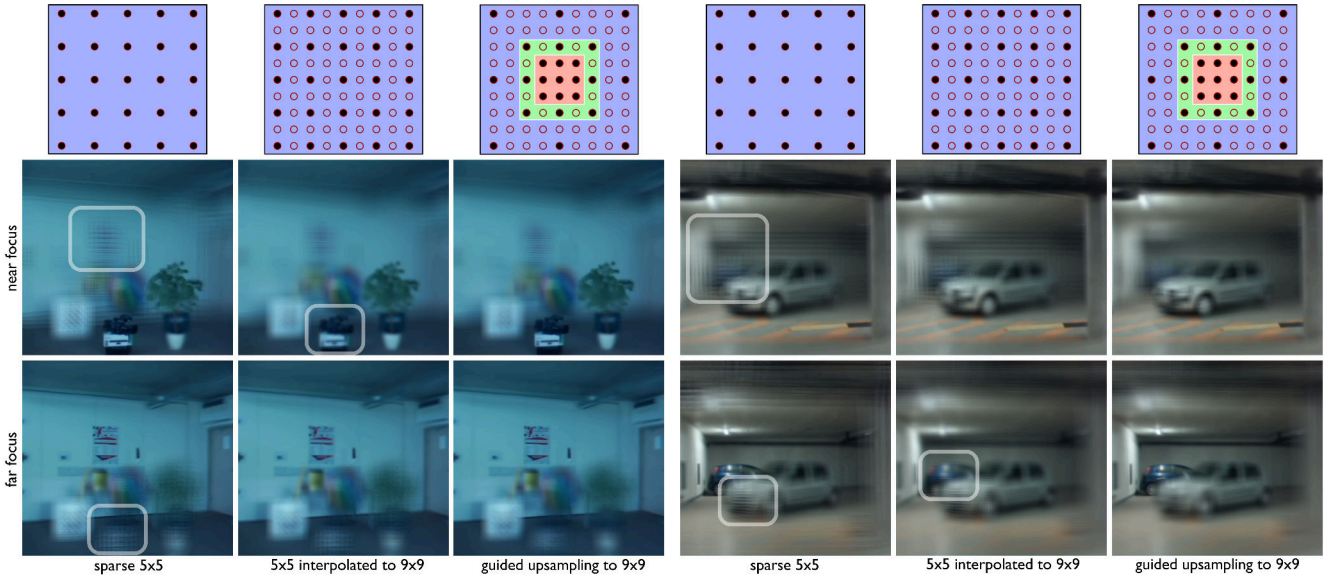

Figure 6. **Synthetic focus rendering.** Two scenes are shown at two focus settings. The scenes were captured with two different alignments of 25 cameras (uniformly spaced and coded, as shown in the top row). The presented renderings compare sparse and interpolated uniformly captured light fields with coded light fields computed with our iterative guided upsampling method.

Figure 7. **Light-field perspectives.** Single light-field perspectives used for image-based rendering, computed by our iterative guided upsampling method (top row) and by interpolation (bottom row). The yellow circles indicate the positions of the perspectives in the sampling patterns.

(e.g., sharp shadows or highly specular surface reflections). Note that the presented results are simulations that approximate the conditions of light stage 6 as closely as possible. For instance, the dimensions of objects, sphere diameter, and LED distances were the same as in reality, the *head* scene was rendered with subsurface scattering, and all renderings were high-dynamic range. In addition to providing visual results, our simulation also allows a quantitative comparison by computing the mean squared error (MSE) in relation to the simulated ground truth. Again, our results indicate that coded sampling and guided upsampling perform significantly better than sparse uniform sampling (i.e., 6V icosahedron with 362 LEDs) or interpolation (e.g., from a 6V to a 12V icosahedron with 1442 LEDs). In figures 9 and 11 we compare several upsampled lighting bases computed by our approach with the corresponding ground truth.

6.1. Heuristically Determined Patch Sizes

We experimentally determined the optimal patch sizes for guided upsampling. For the light stage experiments, we upsampled the scenes with varying patch sizes. Since ground truth data is available in this case, the MSEs can be calculated for each resulting lighting bases image (see table 1). The finally applied patch sizes (15×15 for *head*,

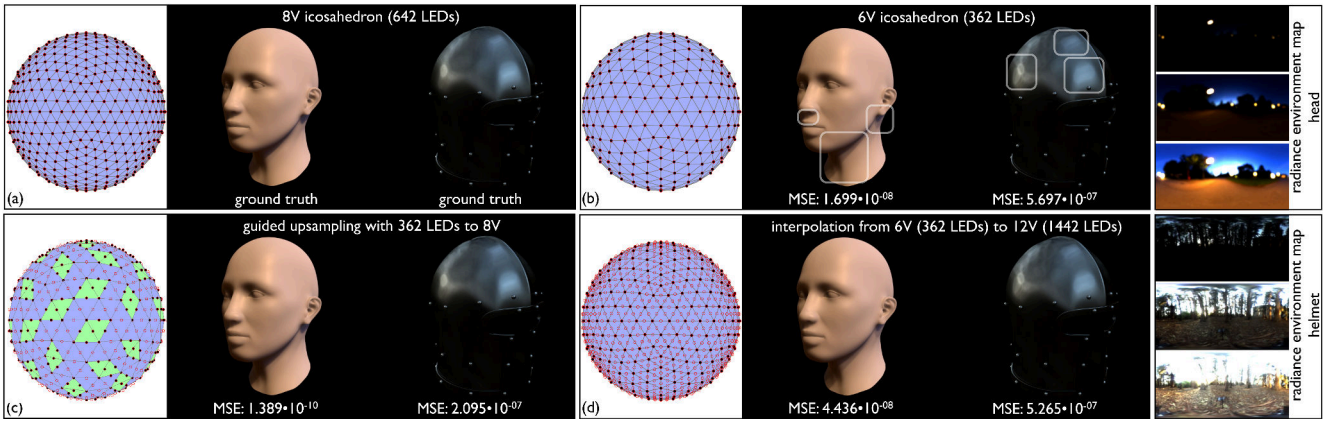

Figure 8. **Image-based relighting.** Two scenes are illuminated with different radiance environment maps (right). (a) The ground truth results achieved with 642 lighting directions. (b) Results achieved with 362 lighting directions (light stage 6), and (c) with our guided upsampling from 362 to 642 lighting directions. (d) Interpolation to the next higher icosahedron subdivision (12V) from 6V. White boxes in (b) highlight typical aliasing artifacts.

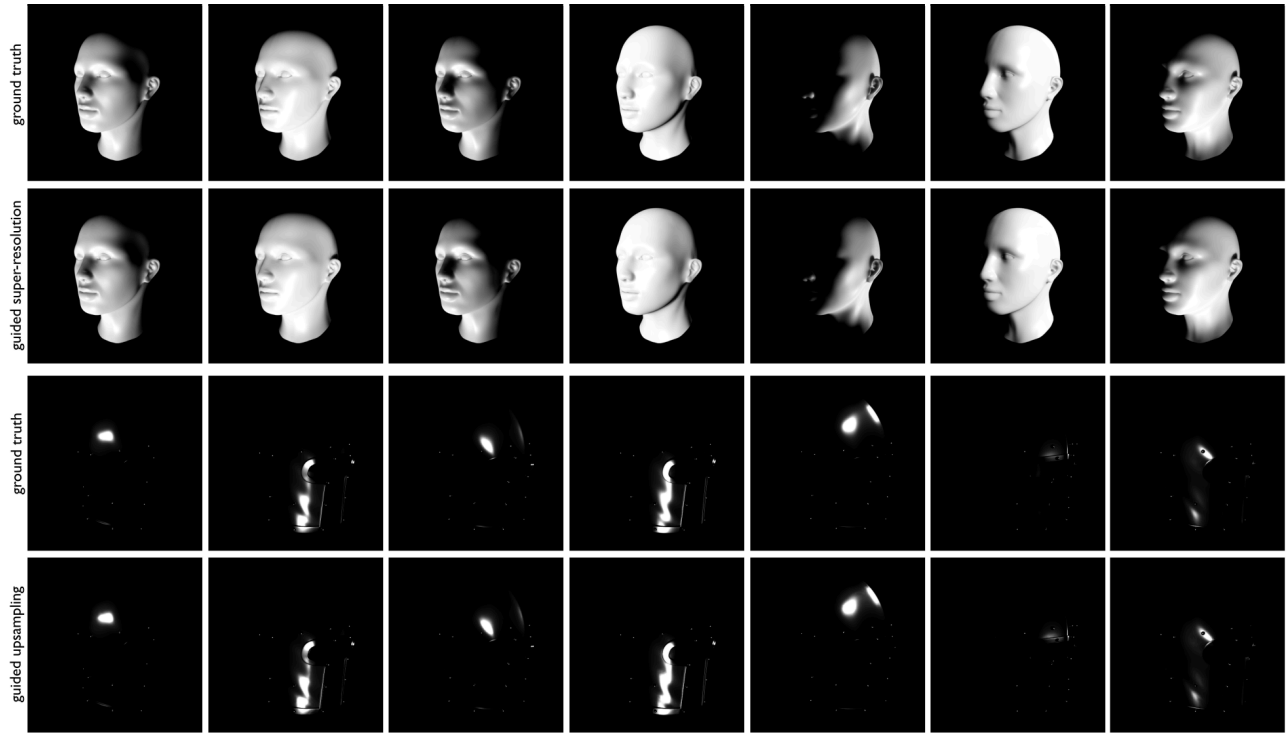

Figure 9. **Lighting bases.** Various guided upsampled lighting bases used for image-based relighting compared with the ground truth bases.

5×5 for *helmet* and 9×9 for *full body*) have been chosen based on the minimal MSE value.

Similarly, we determine the optimal patch sizes for the light field scenes. A ground truth of the recorded light fields for comparison with our results, however, is per se not available. To record at least some ground truth samples for determining the best patch size, we built a camera array as shown in figure 12d. It consists of 49 cameras which are aligned in such a way that light fields with a uniform 5×5 sampling

patch size	head	helmet	full body
$5 \times 5 \times 6$	$3.876{\cdot}10^{-5}$	$\mathbf{1.878}{\cdot}10^{-4}$	$2.880{\cdot}10^{-5}$
$7 \times 7 \times 6$	$3.737{\cdot}10^{-5}$	$2.077{\cdot}10^{-4}$	$2.692{\cdot}10^{-5}$
$9 \times 9 \times 6$	$3.687{\cdot}10^{-5}$	$2.237{\cdot}10^{-4}$	$\mathbf{2.655}{\cdot}10^{-5}$
$11 \times 11 \times 6$	$3.666{\cdot}10^{-5}$	$2.409{\cdot}10^{-4}$	$2.658{\cdot}10^{-5}$
$13 \times 13 \times 6$	$3.611{\cdot}10^{-5}$	$2.551{\cdot}10^{-4}$	$2.688{\cdot}10^{-5}$
$15 \times 15 \times 6$	$\mathbf{3.555}{\cdot}10^{-5}$	$2.688{\cdot}10^{-4}$	$2.720{\cdot}10^{-5}$

Table 1. **Determine patch size for light stage.** MSEs for the light stage scenes *head*, *helmet* and *full body*.

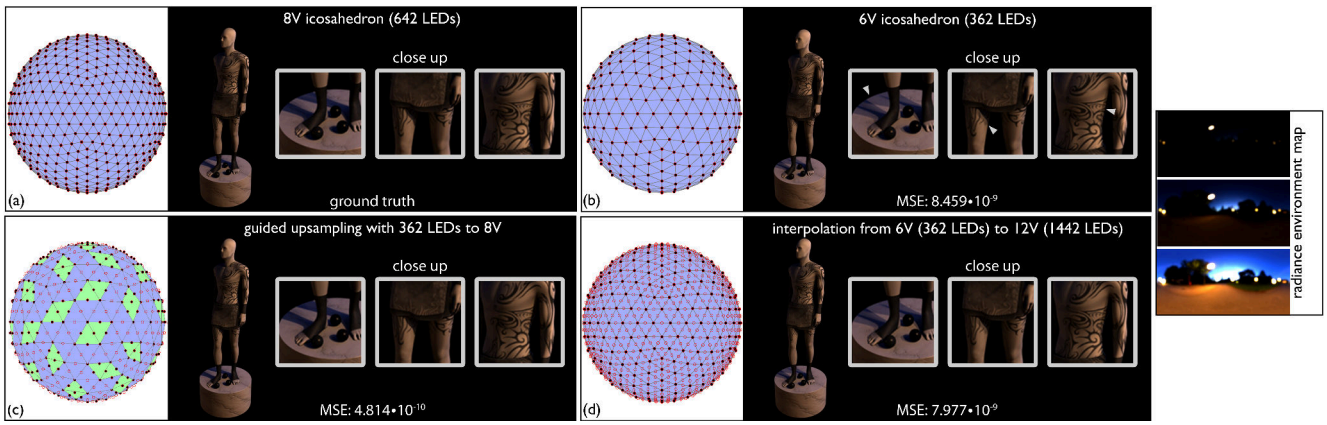

Figure 10. **Image-based relighting.** The *full body* scene is illuminated with a radiance environment map (right). (a) The ground truth achieved with 642 lighting directions. (b) Result achieved with 362 lighting directions (light stage 6), and (c) with our guided upsampling from 362 to 642 lighting directions. (d) Interpolation to the next higher icosahedron subdivision (12V) from 6V. White arrows in (b) highlight typical aliasing artifacts.

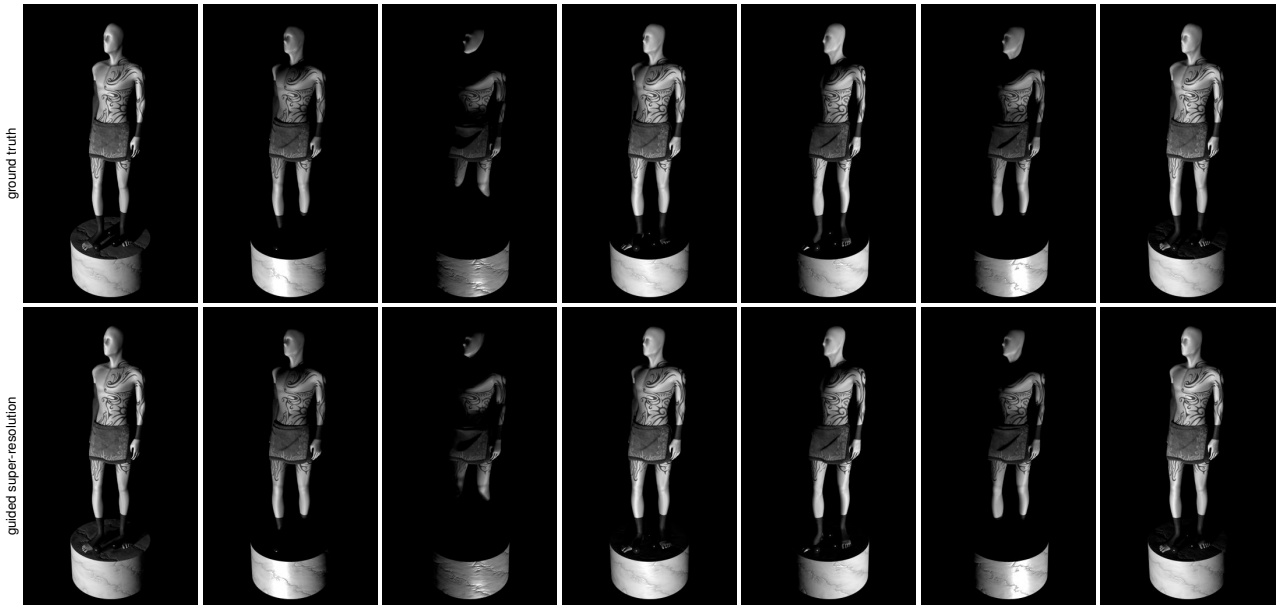

Figure 11. **Lighting bases.** Various guided upsampled lighting bases of the *full body* scene compared with the ground truth bases.

	cars		laboratory	
patch size	iter. 1	iter. 2	iter. 1	iter. 2
$3 \times 3 \times 3 \times 3$	275.52	127.08	141.05	75.92
$5 \times 5 \times 3 \times 3$	212.68	106.89	120.67	**74.28**
$7 \times 7 \times 3 \times 3$	187.20	**102.32**	117.21	74.84
$9 \times 9 \times 3 \times 3$	178.95	103.14	116.81	76.60
$11 \times 11 \times 3 \times 3$	178.34	103.98	**115.63**	77.45
$13 \times 13 \times 3 \times 3$	173.35	105.04	116.28	78.39
$15 \times 15 \times 3 \times 3$	**166.76**	105.88	116.90	79.39
$21 \times 21 \times 3 \times 3$	170.12	106.41	118.16	79.78

Table 2. **Determine patch size for light field.** Averaged MSEs of upsampled perspectives for the light field scenes *cars* and *laboratory* over two iterations.

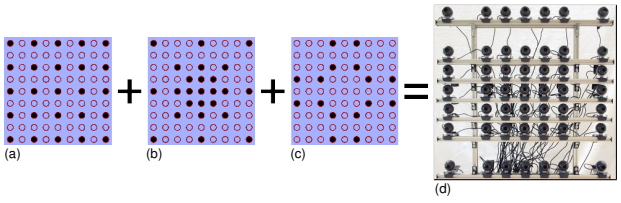

Figure 12. **Camera array.** (d) Our camera array consists of 49 cameras and simultaneously records light fields with (a) a uniform 5×5 and (b) our coded sampling. (c) 16 additional views belong to the desired 9×9 resolution.

(figure 12a) and with our coded alignment (figure 12b) can be captured simultaneously. This requires 33 cameras in total. Furthermore, we add 16 additional views (figure 12c) that belong to the desired 9×9 resolution at selected sampling positions (for first and second iterations). These additional views (figure 12a and figure 12c) are only used as ground truth perspectives for estimating the best patch size and for evaluation. Only the code shown in 12b is used for upsampling. Table 2 presents the resulting MSE values for the *cars* (15×15 and 7×7) *laboratory* (11×11 and 5×5) scenes over the two iterations. We chose the patch sizes with minimal MSE.

Our experiments show that the varying optimal patch sizes depend on the disparity ranges of the scene. The *laboratory* scene, for example, has less depth variation (distance between the focus plane of the foremost objects and the background objects) when compared to the *cars* scene. Therefore, smaller patch sizes are sufficient for upsampling in this case.

7. Limitations and Future Work

We have presented a simple guided super-resolution technique for increasing directional resolution that—rather than relying on depth estimation, disparity computations, or exact image correspondences—searches for best-matching multidimensional (4D or 3D) patches within the entire captured data set to compose new directional images that are consistent in both the spatial and the directional domains. We have described algorithms for guided upsampling, iterative guided upsampling, and sampling code estimation. Our experimental results reveal that the outcomes of existing light-field cameras and light-stage systems can be improved without additional hardware requirements or recording effort simply by changing their sampling patterns by realigning cameras or light-sources. The greatest limitation of our directional super-resolution approach is that it fails if the disparity among neighboring samples is too large. This happens if cameras or light sources are too far apart or their distance to the scene is too short. In this case, good patch matches are difficult to find. Our experiments revealed that disparities larger than 10 pixels for image resolutions of 512×512 are unsuitable for our current patch-matching method. Figure 13 illustrates these limitation based on the example of a 1D camera array (a 180 degree TimeTrack array with 49 cameras used for spatio-temporal blending effects in cinematography and advertising [18]). We used only 29 of the 49 camera perspectives for guided upsampling with the sampling code shown in figure 13. The re-

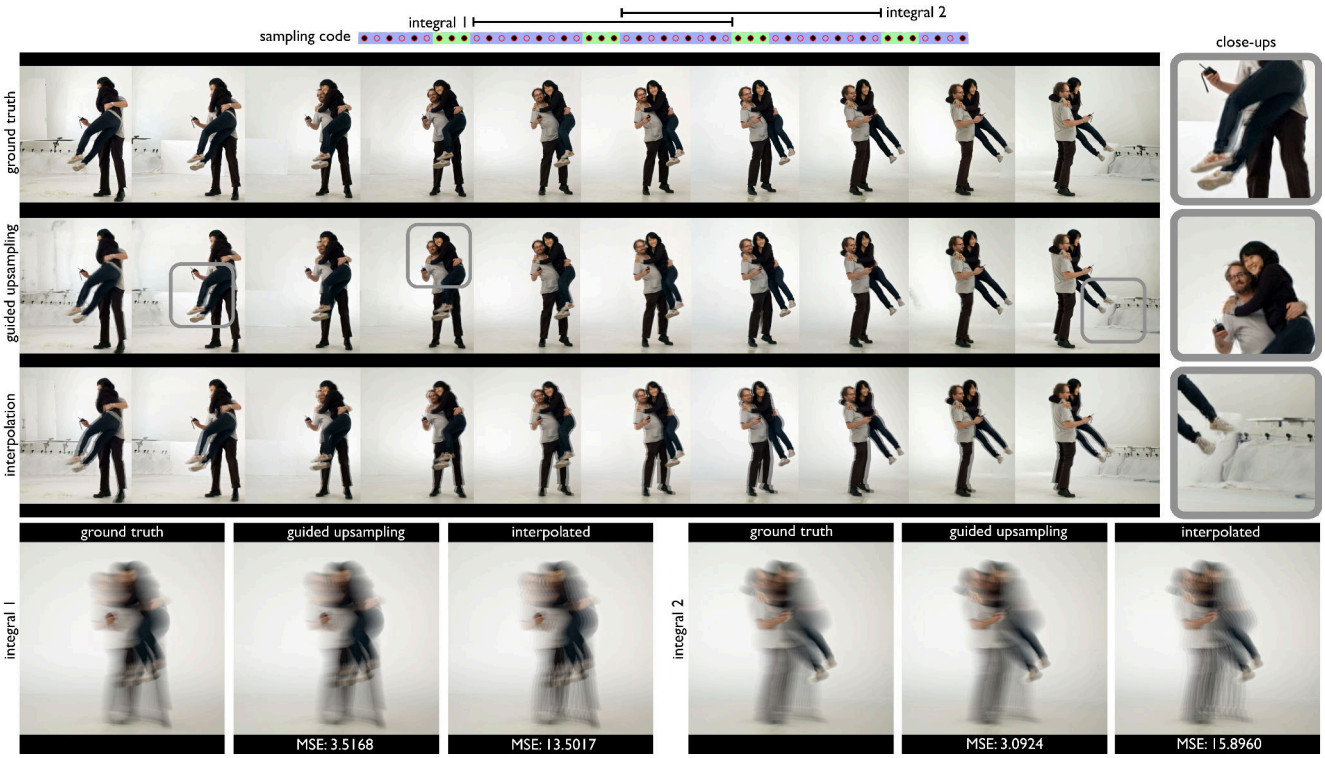

Figure 13. **One-dimensional camera array.** Twenty perspectives of a 1D (180 degree) camera array are interpolated / upsampled with guidance (sampling code shown at top) to 49 images. Example results and the corresponding ground truth images are shown in the first three rows. Close-ups of guided upsampling results are shown on the right. Two integral images (integrated perspectives are indicated at the sampling code) that are computed from the ground truth, the interpolated, and the guided upsampled data are shown at the bottom.

scene	target resolution	memory	time
head	$512 \times 512 \times 642$	15.0 GB	18:53
helmet	$512 \times 512 \times 642$	2.6 GB	4:02
full body	$512 \times 512 \times 642$	6.4 GB	9:13
cars	$520 \times 480 \times 9 \times 9$	12.3 GB	0:18
laboratory	$520 \times 480 \times 5 \times 5$	7.3 GB	0:11

Table 3. **Benchmarks.** Memory consumption and the computation times (h:min) for the light stage scenes *head*, *helmet* and *full body* and the light field scenes *cars* and *laboratory*.

maining 20 perspectives served as ground truth. Note that the patches to be matched were 3D rather than 4D in this example. Compared to linear interpolation, our upsampling results are better. However, they remain insufficient for virtual camera movements (frozen-moment camera movement effect) where only a single image is shown at a time. For synthetic focus rendering of light fields and image-based relighting of light-stage data, minor patch-matching artifacts are less critical, as they are usually averaged by blending multiple images for the final result. This is also illustrated in figure 13 with the integral images computed from multiple perspectives. Another limitation is the computational complexity and thus the memory requirements of patch matching (see table 3). Even with a fast approximate nearest neighbor search, as is implemented in FLANN, processing time and memory usage for the presented results were 5–9 minutes per iteration and 7.3–12.3 GB of peak memory (for $520 \times 480 \times 9 \times 9$ light fields), and 4–19 hours and 2.6–15 GB (for a light stage with 642 lighting bases and an image resolution of 512×512 pixels). All computations were carried out on an Intel Core i5 3.2 GHz with 16 GB RAM.

In the future, we intend to investigate better patch-matching techniques that allow larger disparities. These will support sparser capturing systems and the use of individual (non-blended) output images, such as required for frozen-moment camera movement effects. Furthermore, we are planning to extend our approach to the temporal domain. Multiple time frames will enlarge the source of matchable 4D/3D patches and can consequently enhance the quality of single frames. To ensure temporal consistency, however, our method needs to be extended to 5D/4D patches.

References

[1] T. E. Bishop, S. Zanetti, and P. Favaro. Light field superresolution. In *ICCP*, 2009.

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

[3] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum. Plenoptic sampling. In *SIGGRAPH*, pages 307–318, 2000.

[4] P. Einarsson, C.-F. Chabert, A. Jones, W.-C. Ma, B. Lamond, T. Hawkins, M. Bolas, S. Sylwan, and P. Debevec. Relighting human locomotion with flowed reflectance fields. In *EGSR*, 2006.

[5] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Trans. Graph.*, 28(3):1–10, 2010.

[6] M. Fuchs, H. P. A. Lensch, V. Blanz, and H. Seidel. Superresolution reflectance fields: Synthesizing images for intermediate light directions. *Comput. Graph. Forum*, 26(3):447–456, 2007.

[7] T. Georgiev, G. Chunev, and A. Lumsdaine. Superresolution with the focused plenoptic camera. In *SPIE Electronic Imaging*, 2011.

[8] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009.

[9] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH*, page 43–54, 1996.

[10] A. Levin and F. Durand. Linear view synthesis using a dimensionality gap light field prior. In *CVPR*, pages 1–8, 2010.

[11] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996.

[12] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar. Compressive Light Field Photography using Overcomplete Dictionaries and Optimized Projections. *ACM Trans. Graph.*, 32(4):1–11, 2013.

[13] V. Masselus, P. Peers, P. Dutr, and Y. D. Willems. Smooth reconstruction and compact representation of reflectance functions for image-based relighting. In *EGSR*, pages 287–298, 2004.

[14] K. Mitra, O. Cossairt, and A. Veeraraghavan. Can we beat hadamard multiplexing? data driven design and analysis for computational imaging systems. In *ICCP*, pages 1–9, May 2014.

[15] K. Mitra and A. Veeraraghavan. Light field denoising, light field superresolution and stereo camera based refocussing using a gmm light field patch prior. In *CVPRW*, pages 22–28, 2012.

[16] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *PAMI*, 36, 2014.

[17] O. Shahar, A. Faktor, and M. Irani. Super-resolution from a single video. In *CVPR*, 2011.

[18] D. Taylor. Virtual camera movement: The way of the future? *American Cinematographer*, 77:93–100, September 1996.

[19] B. Tunwattanapong, A. Ghosh, and P. Debevec. Practical image-based relighting and editing with spherical-harmonics and local lights. In *CVMP*, 2011.

[20] S. Wanner and B. Goldluecke. Spatial and angular variational super-resolution of 4d light fields. In *ECCV*, 2012.

[21] B. Wilburn, M. Smulski, H.-H. K. Lee, and M. Horowitz. The light field video camera. In *Media Processors, SPIE Electronic Imaging*, 2002.