Supplementary Material - Deep Material Recognition in Light-Fields via Disentanglement of Spatial and Angular Information

Bichuan Guo[1[0000−0001−8475−427X]], Jiangtao Wen[1], and Yuxing Han[2]

[1] Tsinghua University, Beijing, China
[2] Research Institute of Tsinghua University in Shenzhen, Shenzhen, China
gbc16@mails.tsinghua.edu.cn

In this supplementary material we provide (1) more technical details about angular registration and some visualizations of its effect; (2) the detailed network architecture of our proposed angular feature extractors *angular-4* and *angular-S*, as well as an analysis of the angular rotational invariance property; (3) more visual results similar to Sec. 5.4 in the main paper.

S.1 Angular registration

We provide more details on the implementation of the proposed angular registration algorithm (Sec. 4.2), as well as some visual examples to demonstrate its performance.

S.1.1 Implementation

The discrete Fourier transform of a sub-view is computed via the 2D fast Fourier transform `fft2` [*S7*]. Many `fft2` software implementations put the zero-frequency component in the low-order corner of the spectrum [*S1*, *S2*]. In this case, we need to shift the zero-frequency component to the center of the spectrum, this can be done by calling `fftshift` in MATLAB or `numpy` [*S3*].

Since the phase of the cross power spectrum $\arg \mathcal{S}$ is periodic, the median filter must take this periodicity into account. For a set containing phase data $A = \{\theta_1, \theta_2, ..., \theta_n\}$, the circular median [*S4*] of A is defined as

$$\text{med}[A] = \underset{\theta_i \in A}{\text{argmin}} \sum_{j=1}^{n} \|\theta_i - \theta_j\|_c, \tag{S1}$$

where we compute the circular distance $\|\theta\|_c$ of an angle θ by first converting θ to the range $[-\pi, \pi)$, then taking its absolute value. The median filter replaces a pixel p with the circular median of all pixels that are within a radius of 3 pixel units centered at p, in Euclidean distance.

Since we have shifted the zero-frequency component to the center of the spectrum, we can regress $\arg \mathcal{S}_\xi$ on $(\xi/S - 0.5)$ without an intercept, this leaves only one coefficient (the slope) to be determined.

S.1.2 Visual examples

Here we provide some visual examples to illustrate angular registration. We display the 4D light-field in lenslet format [*S5*], that is, we replace the pixel p in the center sub-view with a $U \times V$ block of pixels from all sub-views, which all have the same spatial coordinates as p, and $U \times V$ is the angular resolution. The lenslet format light-field is then a 2D image with resolution $(S \times U) \times (T \times V)$, where $S \times T$ is the spatial resolution. This is the same format as the raw data of the light-field material dataset [*S6*].

We show several examples of the light-field before and after angular registration in Fig. S1, S2 and S3. It is a common pattern that before angular registration, the pixels within a $U \times V$ block often correspond to different surface points, which is especially apparent when near edges. After angular registration, sub-views are much better aligned, pixels within the same $U \times V$ block can better represent the reflection variation of a single surface point.

We stress that this alignment is not done via dense depth estimation, but using **a single scalar value**: the disparity between adjacent sub-views, estimated from the frequency domain. The ability to align all sub-views using a single scalar is a strong evidence for the geometric assumptions about the spatial and angular domains: the disparity linearity (2), approximately constant depth of the representative region, and the square grid model of the micro-lenses array.

S.2 Angular feature extractor

We provide the detailed network architectures of the angular feature extractors *angular-4* and *angular-S*, and a detailed discussion on *angular-S*'s angular rotational invariance property.

S.2.1 Network architecture

layer name	type	kernel size	padding	dilation	in channel	out channel
af	Conv2D	K_{af}	$(K_{\text{af}} - 1)/2$	1	21	#af
relu_1_1	ReLU	-	-	-	-	-
conv_1_1	Conv2D	3	1	1	#af	#af
bn_1	BatchNorm	-	-	-	-	-
relu_1_2	ReLU	-	-	-	-	-

Table S1. Architecture of the angular filter sub-network. Here $K_{\text{af}} \in \{1, 3\}$ and $\#\text{af} \in \{16, 32, 64\}$ are hyper-parameters. The listed parameters apply to convolutional layers but not ReLU or batch normalization layers (indicated by "-"). Batch normalization layers use $\epsilon = 10^{-5}$ and a momentum of 0.1.

layer name	type	kernel size	padding	dilation	in channel	out channel
conv_2_1	Conv2D	3	1	1	4 * #af	4 * #af
relu_2_1	ReLU	-	-	-	-	-
conv_2_2	Conv2D	3	1	1	4 * #af	4 * #af
bn_2	BatchNorm	-	-	-	-	-
relu_2_2	ReLU	-	-	-	-	-
pool_2	MaxPool2D	2	0	2	4 * #af	4 * #af
conv_3_1	Conv2D	3	1	1	4 * #af	8 * #af
relu_3_1	ReLU	-	-	-	-	-
conv_3_2	Conv2D	3	1	1	8 * #af	8 * #af
bn_3	BatchNorm	-	-	-	-	-
relu_3_2	ReLU	-	-	-	-	-
pool_3	MaxPool2D	2	0	2	8 * #af	8 * #af
conv_4_1	Conv2D	3	1	1	8 * #af	16 * #af
relu_4_1	ReLU	-	-	-	-	-
conv_4_2	Conv2D	3	1	1	16 * #af	16 * #af
bn_4	BatchNorm	-	-	-	-	-
relu_4_2	ReLU	-	-	-	-	-
pool_5	AveragePool2D	global	0	-	16 * #af	16 * #af

Table S2. Architecture of the shared convolutional network for *angular-4*. Here #af $\in \{16, 32, 64\}$ is a hyper-parameter. The listed parameters apply to convolutional layers and pooling layers, but not ReLU or batch normalization layers (indicated by "-"). Batch normalization layers use $\epsilon = 10^{-5}$ and a momentum of 0.1. The final layer is a global average pooling layer, i.e. the kernel size is taken to be equal to the input size, indicated by "global".

layer name	type	kernel size	padding	dilation	in channel	out channel
direct_pool	MaxPool3D	(1, 1, 4)	0	(1, 1, -)	(#af, W, H, 4)	(#af, W, H)
conv_2_1	Conv2D	3	1	1	#af	#af
relu_2_1	ReLU	-	-	-	-	-
conv_2_2	Conv2D	3	1	1	#af	#af
bn_2	BatchNorm	-	-	-	-	-
relu_2_2	ReLU	-	-	-	-	-
pool_2	MaxPool2D	2	0	2	#af	#af
conv_3_1	Conv2D	3	1	1	#af	2 * #af
relu_3_1	ReLU	-	-	-	-	-
conv_3_2	Conv2D	3	1	1	2 * #af	2 * #af
bn_3	BatchNorm	-	-	-	-	-
relu_3_2	ReLU	-	-	-	-	-
pool_3	MaxPool2D	2	0	2	2 * #af	2 * #af
conv_4_1	Conv2D	3	1	1	2 * #af	4 * #af
relu_4_1	ReLU	-	-	-	-	-
conv_4_2	Conv2D	3	1	1	4 * #af	4 * #af
bn_4	BatchNorm	-	-	-	-	-
relu_4_2	ReLU	-	-	-	-	-
pool_5	AveragePool2D	global	0	-	4 * #af	4 * #af

Table S3. Architecture of the shared convolutional network for *angular-S*. The notation for the directional pooling layer needs to be clarified: the triplets of "kernel size" and "dilation" correspond to input width W, input height H and the new axis "direction", respectively. The subsequent layers are the same as *angular-4*, except that the numbers of input and output channels are reduced by 4.

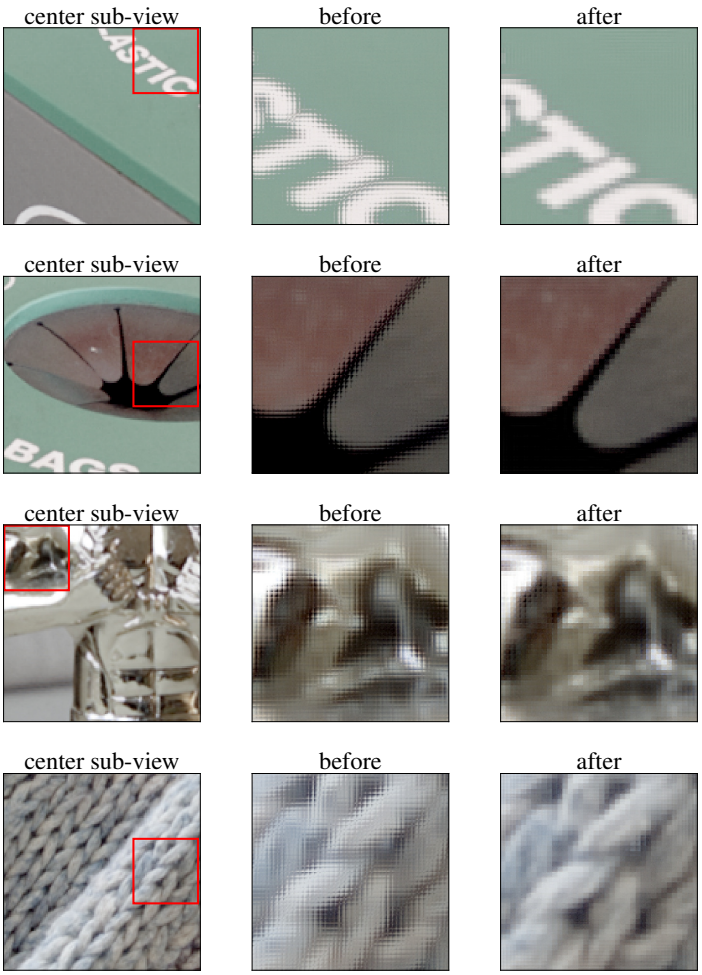

Fig. S1. Illustration of the angular registration algorithm (part 1). The representative region is outlined in red in the center sub-view (left). Before angular registration (middle), pixels with same spatial coordinates correspond to different surface points. This is easily observed when near edges. After angular registration (right), sub-views are much better aligned, pixels within the same $U \times V$ block can better represent the reflection variation of a single surface point. Zoom in for better observation.

In our proposed angular feature extractors *angular-4* and *angular-S*, four stacks of sub-views first pass through angular filter sub-networks, which produce four feature maps, one for each stack. The detailed architecture of the angular filter sub-network is shown in Table S1. Both *angular-4* and *angular-S* use the same architecture, the only difference is that, *angular-4* uses four dif-

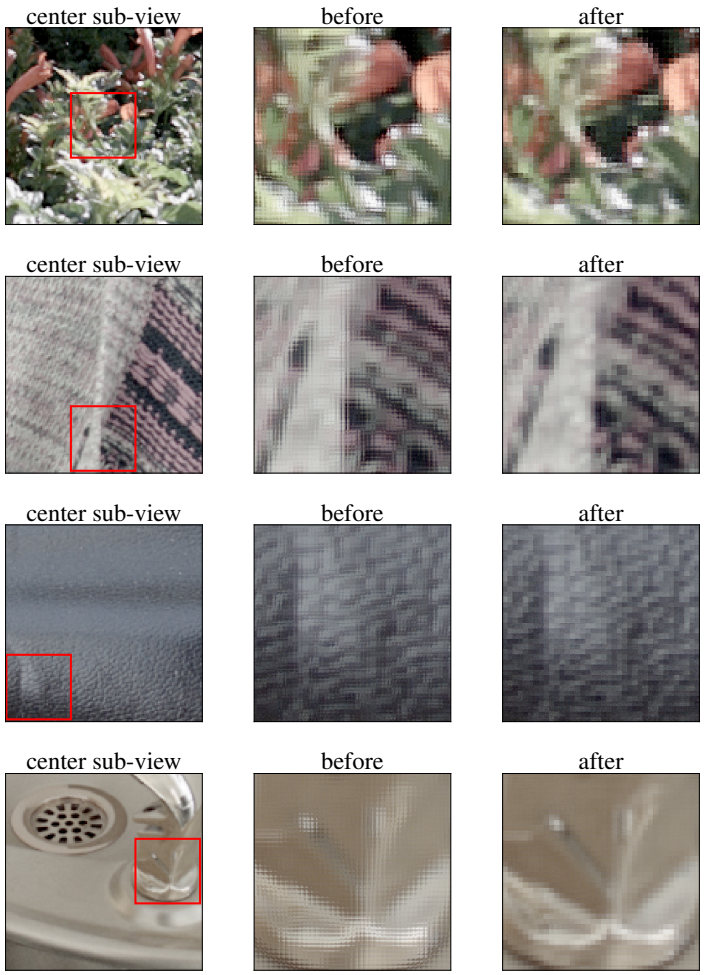

Fig. S2. Illustration of the angular registration algorithm (part 2). The representative region is outlined in red in the center sub-view (left). Before angular registration (middle), pixels with same spatial coordinates correspond to different surface points. This is easily observed when near edges. After angular registration (right), sub-views are much better aligned, pixels within the same $U \times V$ block can better represent the reflection variation of a single surface point. Zoom in for better observation.

ferent angular filter sub-networks, one for each sub-view stack, while *angular-S* uses the same angular filter sub-network for all stacks, such that layer weights are shared.

With four feature maps produced by angular filter sub-networks, each with #af channels, *angular-4* concatenates four feature maps depth-wise. The con-

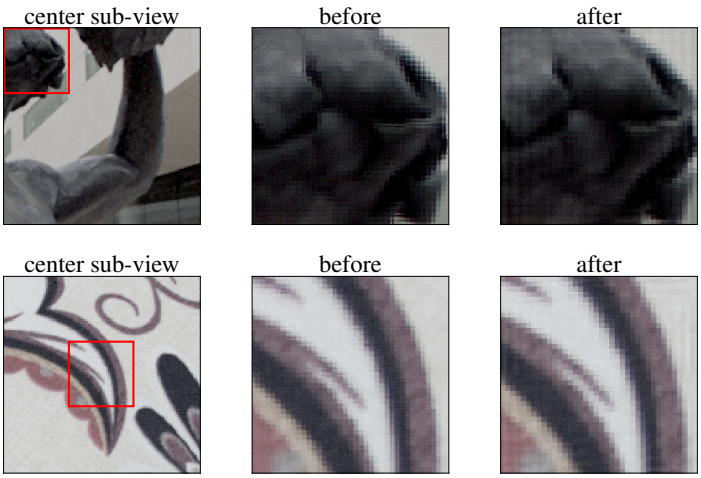

Fig. S3. Illustration of the angular registration algorithm (part 3). The representative region is outlined in red in the center sub-view (left). Before angular registration (middle), pixels with same spatial coordinates correspond to different surface points. This is easily observed when near edges. After angular registration (right), sub-views are much better aligned, pixels within the same $U \times V$ block can better represent the reflection variation of a single surface point. Zoom in for better observation.

catenated tensor, with 4×#af channels, is fed to a shared convolutional network, the detailed architecture of which is shown in Table S2. The final layer is a global average pooling layer, i.e. the kernel size is taken to be equal to the input size, indicated by "global".

Comparing to *angular-4*, *angular-S* uses a slightly different shared convolutional network, shown in Table S3. Four feature maps from the angular filter sub-network are not concatenated along the depth axis, instead they are concatenated along a new "direction" axis, making the new feature map a 4D structure. For example, if the feature maps from the angular filter sub-network are of shape #af $\times W \times H$, the resulting tensor will be of size #af $\times W \times H \times 4$. We first apply the directional pooling layer to this new axis (the last axis in this example), which is a max-pooling layer that selects the largest element among four feature maps. This produces a feature map with size back to #af $\times W \times H$. The subsequent layers are the same as *angular-4*, except that the numbers of input and output channels are reduced by 4, since we have used directional pooling to reduce the amount of data by 4.

S.2.2 Angular rotational invariance

We give a more detailed analysis of the angular rotational invariance property of *angular-S*. Denote four sub-view stacks by s_1, s_2, s_3 and s_4, and the function

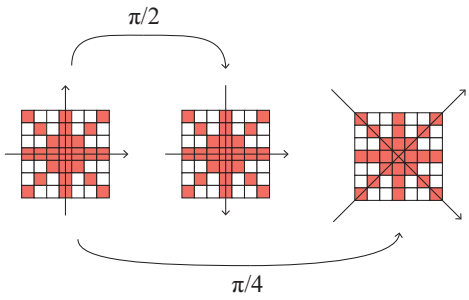

Fig. S4. Illustration of the angular rotational invariance property, where the angular domain axes rotate clockwisely by $\pi/2$ or $\pi/4$. If the axes rotate by $\pi/2$, the set of sub-view stacks $\{s_1, s_2, s_3, s_4\}$ is exactly a permutation of the original set. If the axes rotate by $\pi/4$, the integral coordinate points are not exactly the same as the original, but the overall structure still maintains the crosshair pattern and can be regarded as a good approximation to the permutation of the original set.

represented by the angular filter sub-network by $g(\cdot)$. The output of the directional pooling layer is an element-wise maximum $\max_{i\in\{1,2,3,4\}} g(s_i)$. We see that upon permutation of the set $\{s_1, s_2, s_3, s_4\}$, the output of the directional pooling layer is invariant.

If the camera or object rotates, the angular domain axes will also rotate. Fig. S4 shows the scenarios where the angular domain axes rotate by $\pi/2$ or $\pi/4$. If the axes rotate by $\pi/2$, the set of sub-view stacks $\{s_1, s_2, s_3, s_4\}$ is exactly a permutation of the original set. If the axes rotate by $\pi/4$, the integral coordinate points are not exactly the same as the original, but the overall structure still maintains the crosshair pattern and can be regarded as a good approximation to the permutation of the original set.

S.3 More visualization results

We provide more examples on visualization (Sec. 5.4) here due to the space limit of the original paper.

S.3.1 Angular filter output

Fig. S5 and S6 provide more visual examples similar to Fig. 6 in the original paper. Fig. S5 shows the angular filter activations of sub-view stacks for non-Lambertian materials. We see the activations of different sub-view stacks differ, indicating the materials have strong reflection variations. Fig. S6 shows the angular filter activations of sub-view stacks for materials with homogeneous reflection. We see the activations are similar across directions. This confirms our claim in the original paper.

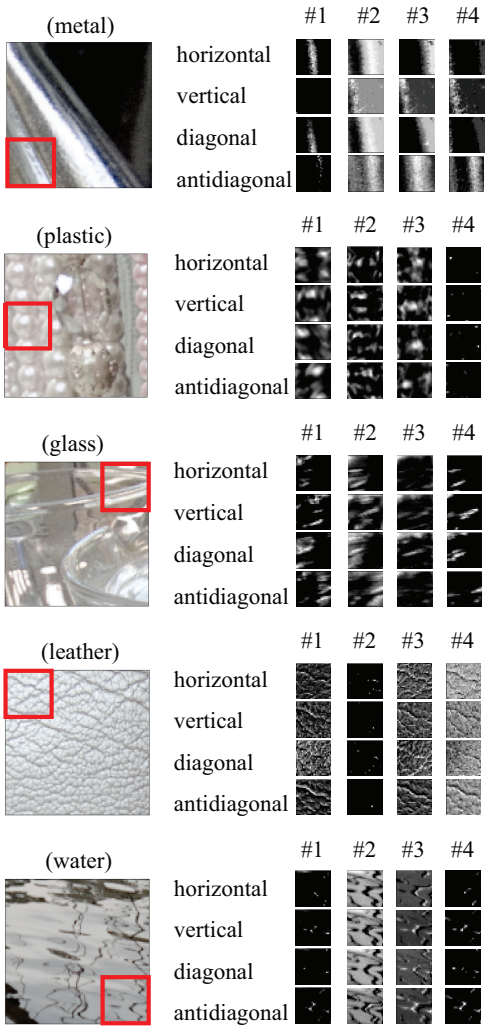

Fig. S5. Angular filter activations of sub-view stacks for non-Lambertian materials. Left: Center sub-views of the original light-field patches, representative regions are outlined in red. Right: each column is a feature map from the angular filter of *angular-S*, each row corresponds to a stack of sub-views. The activations of different sub-view stacks differ, indicating the materials have strong reflection variations.

S.3.2 Angular v.s. spatial responses

Fig. S7, S8 and S9 provide more visual examples similar to Fig. 6 in the original paper. We also observe the pattern that, if the patch contains object or shape information so that the material can be easily inferred, the spatial response is

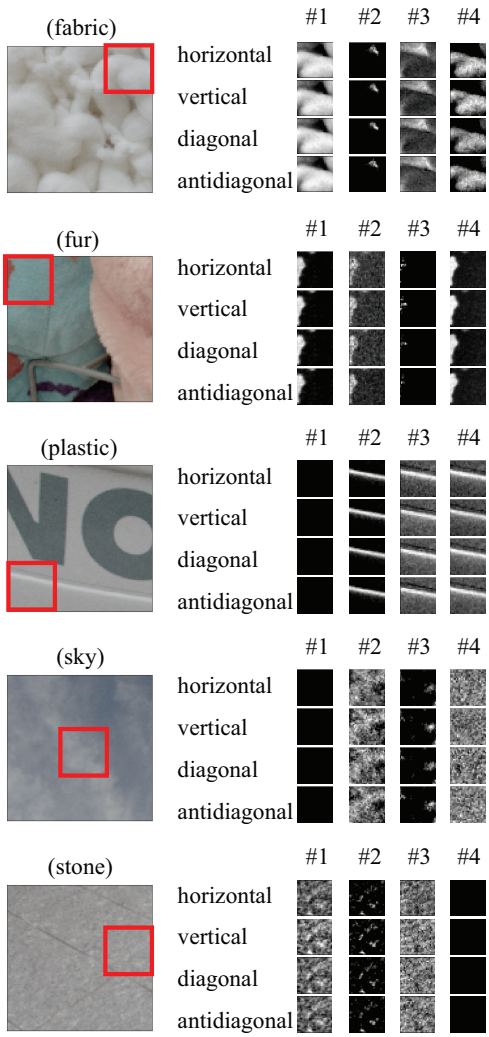

Fig. S6. Angular filter activations of sub-view stacks for materials with homogeneous reflection. Left: Center sub-views of the original light-field patches, representative regions are outlined in red. Right: each column is a feature map from the angular filter of *angular-S*, each row corresponds to a stack of sub-views. Activations are similar across directions.

much higher than angular response. Conversely, if context information is vague, the angular response becomes more significant. This confirms our claim in the original paper.

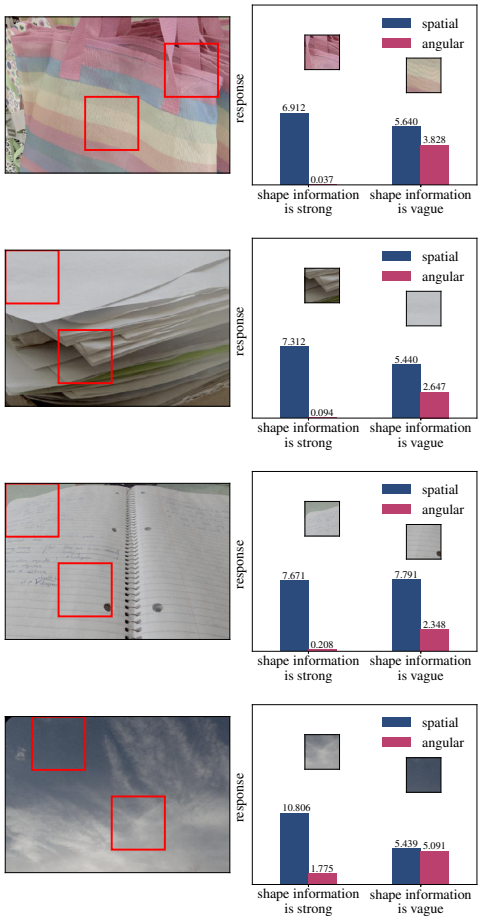

Fig. S7. Angular/spatial responses of different patches in the same light-field image (part 1). Left: the center sub-view of the original light-field image, selected patches are outlined in red. Right: corresponding patches are displayed above bar graphs. When shape information is vague, angular response becomes more significant, and vise versa.

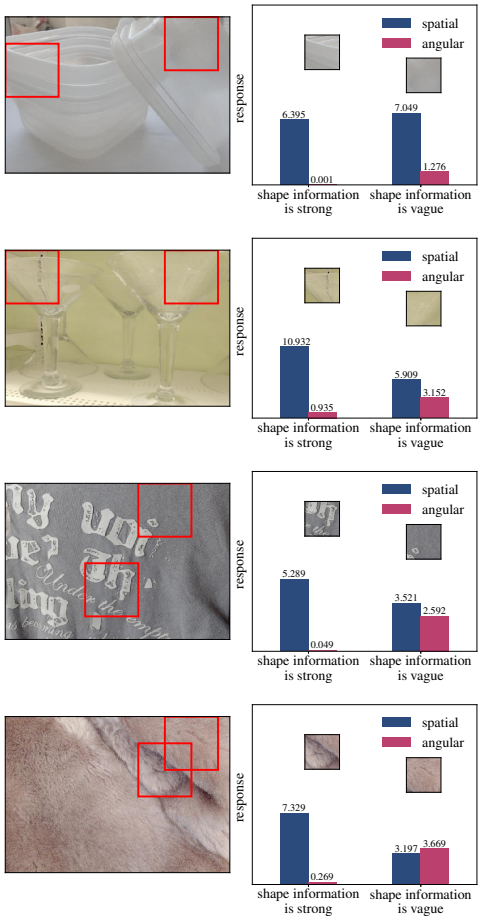

Fig. S8. Angular/spatial responses of different patches in the same light-field image (part 2). Left: the center sub-view of the original light-field image, selected patches are outlined in red. Right: corresponding patches are displayed above bar graphs. When shape information is vague, angular response becomes more significant, and vise versa.

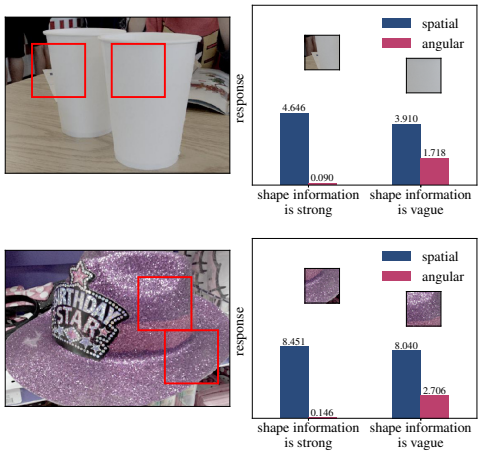

Fig. S9. Angular/spatial responses of different patches in the same light-field image (part 3). Left: the center sub-view of the original light-field image, selected patches are outlined in red. Right: corresponding patches are displayed above bar graphs. When shape information is vague, angular response becomes more significant, and vise versa.

Bibliography

[S1] 2-D fast Fourier transform. https://www.mathworks.com/help/matlab/ref/fft2.html, accessed: 2019-11-01

[S2] numpy.fft.fft2. https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft2.html, accessed: 2019-11-01

[S3] numpy.fft.fftshift. https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fftshift.html, accessed: 2019-11-01

[S4] Storath, M., Weinmann, A.: Fast median filtering for phase or orientation data. IEEE TPAMI **40**(3) (2018)

[S5] Vieira, A., Duarte, H., Perra, C., Tavora, L., Assuncao, P.: Data formats for high efficiency coding of lytro-illum light fields. In: 2015 International Conference on Image Processing Theory, Tools and Applications (2015)

[S6] Wang, T.C., Zhu, J.Y., Hiroaki, E., Chandraker, M., Efros, A.A., Ramamoorthi, R.: A 4D light-field dataset and CNN architectures for material recognition. In: ECCV (2016)

[S7] Weisstein, E.W.: Fast Fourier transform (2015)