Semantic See-Through Rendering on Light Fields

Huangjie Yu∗ Guli Zhang∗ Yuanxi Ma∗ Yingliang Zhang∗ Jingyi Yu∗

Abstract
We present a novel semantic light field (LF) refocusing technique that can achieve unprecedented see-through quality. Different from prior art, our semantic see-through (SST) differentiates rays in their semantic meaning and depth. Specifically, we combine deep learning and stereo matching to provide each ray a semantic label. We then design tailored weighting schemes for blending the rays. Although simple, our solution can effectively remove foreground residues when focusing on the background. At the same time, SST maintains smooth transitions in varying focal depths. Comprehensive experiments on synthetic and new real indoor and outdoor datasets demonstrate the effectiveness and usefulness of our technique.

CCS Concepts
• Computing methodologies → Computer vision tasks; Image-based rendering; Visibility;

Keywords
Image-based rendering, light field, lumigraph, ray space analysis, semantic segmentation, synthetic aperture, depth of field

1 INTRODUCTION
A plenoptic function describes rays originating from any point and along any direction in free space [2]. A practical implementation of the plenoptic function is the light field [21], widely used for image-based modeling and rendering and most recently for virtual and augmented reality [23, 28]. A LF adopts the two plane parametrization (2PP): each ray is parameterized by its intersections of two parallel planes \( st \) and \( uv \) at \( z = 0 \) and \( z = 1 \) respectively as a 4-tuple \( r = [s, t, u, v] \). Under the camera array setting, \( [s, t] \) can be viewed the camera position and \( [u, v] \) as the pixel coordinate in the respective camera [39, 42].

A unique rendering capability of LF is post-capture refocusing [17, 26]: given a virtual focal plane \( z = d_f \), rays from an LF are resembled as if they emit from the focal plane:

\[
E(u', v') = \int \int L(s, t, s + \frac{u' - s}{d_f}, t + \frac{v' - t}{d_f}) A(s, t) ds dt
\]

where \( A \) represents the virtual aperture that controls the angular extent of rays to gather. The refocusing process can be further accelerated in the frequency space [27]. More recent approaches further employed depth-guided ray interpolation to minimize visual artifacts such as aliasing [1] and color bleeding [9], where the depth map can be acquired either by active 3D sensing [25] or passive light field stereo matching [18, 37].

By setting the aperture really big (e.g., using a LF camera array), the refocusing effect can further mimic virtual see-through [17, 40]. Fig.1 (b) shows that by focusing on the asset hiding behind the bushes, we can partially remove the foreground. However, even at a full aperture (i.e., using all cameras in the array), the rendering still exhibits substantial foreground residue.
This 4D representation enables direct ray space interpolation for process is further accelerated using Fourier space slicing [26] on the depth of the focal plane, they can produce a focal stack. The through an aperture and converging on a focal plane. By changing spling. [17] demonstrated refocusing on LFs by blending rays passing Chai et al. [5] analyzed the relationship between aliasing and sampled views while guiding ray interpolation with geometry proxies.

LF model as a practical representation to the Plenoptic function. In a similar vein, Lumigraph uses irregularly sampled views and volumetric reconstruction [20, 33] to tracking [3, 8, 32]. By far, nearly existing techniques are developed for 2D images rather than structure-from-motion for fusing stereo matching results from multiple LFs. These techniques are suitable for LFs of small baselines (e.g., the Lytro data). For see-through effects, it is desirable to use LFs of very large baselines.

Semantic Labeling. Latest semantic labeling techniques exploit rich data and deep learning techniques [7, 11, 46]. The availability of semantic labels has further benefited many traditionally challenging computer vision problems, ranging from stereo matching and volumetric reconstruction [20, 33] to tracking [3, 8, 32]. By far, nearly existing techniques are developed for 2D images rather than a 4D light field. Wanner et al. [37, 38] presented a coherent light field segmentation technique although their technique does not explicitly use semantic information. We seek to assign each ray in the light field with a semantic label and then employ the results to improve the see-through effects. The seminal work of Pyramid Scene Parsing Network (PSPNet) discovers contextual relationship and global information in complex scenes. The results are outstanding on popular benchmarks. However, their results are sensitive to occlusions and brute-force application to individual LF views leads to incoherence.

Semantic-Guided Reconstruction. Finally, our technique also aligns with recent trends on incorporating semantic information in 3D reconstruction. The seminal work by Kundu et al. used a Conditional Random Field (CRF) model to jointly estimate object labeling and voxels occupation [19]. Häne et al. further proposed a joint optimization framework to simultaneously solve for the image segmentation and 3D reconstruction problems. The key observation is that semantic segmentation and geometric cues are complementary [15, 16]. Although effective, the quality of volumetric reconstruction relies
We extend the pyramid scene parsing network (PSPNet) \[46\] to incorporate multi-scale contextual information in pixel-wise label prediction. To coherently conduct semantic labeling to all views in a LF, we design a two-stage deep network pipeline. The first stage employs a PSPNet to generate an initial semantic label map as well as the marginal distributions of every pixel label. Our second stage introduces a high confidence semantic map (HCSM) prediction module that estimates how reliable each pixel label prediction is. The HCSM is then used to filter out unreliable pixels with low confidence values. Here we first briefly review the PSPNet architecture used in our work and then discuss the HCSM generation module.

The PSPNet first computes a convolutional feature map based on the ResNet and dilated convolutions, which has \(\frac{1}{3}\) of the input image size. To encode spatial context information, the network adopts a pyramid feature pooling strategy that generates a four-level feature maps representing global and sub-regional contexts. The pooled context feature maps are then upsampled and concatenated with the original feature map as inputs to predicting a multi-class label distribution for each pixel. The final semantic labeling can be generated by taking the most-probable label configuration for the entire image. The model configuration is illustrated in Fig.3.

Formally, we denote the label space as \(C = \{1, \cdots, C\}\) where \(C\) is the number of semantic classes. At each pixel \(x_i\), which corresponds to a visual ray \([u, v, s, t]\), the PSPNet computes a label distribution \(p(y_i)\) where \(y_i \in C\). We generate the pixel-wise labeling by taking the MAP estimates as \(y_i^* = \arg\max p(y_i)\).

Given the outputs of the PSPNet, we now estimate a confidence score for each pixel’s label prediction. We observe that the marginal label distribution \(p(y_i)\) tends to be more divergent in complex areas, e.g., near the boundary of an object, and more concentrated within an object region. We therefore compute a HCSM in terms of the entropy of the label distribution as follows,

\[
H(x_i) = - \sum_{y_i \in C} p(y_i) \log p(y_i) \quad (2)
\]

The label distribution at each pixel reaches the maximum entropy when each label value shares the same probability, and reaches the minimum when it takes a single label value with probability 1. A large entropy generally indicates that we are not confident at that pixel due to diverse label probabilities.

Given the HCSM, we use a simple thresholding strategy to filter out pixels with unreliable label predictions. Specifically, we consider the initial semantic label prediction \(y_i^*\) confident if the following condition holds,

\[
H(x_i) = - \sum_{y_i \in C} p(y_i) \log p(y_i) < \epsilon_H \quad (3)
\]

where \(\epsilon_H\) is a model parameter controlling the balance between the precision and recall of the remaining pixel labeling.

To estimate the parameter \(\epsilon_H\), we introduce a score function based on the quality of the label predictions satisfying the above condition. Denote the remaining pixel set as \(S_r\), we first use the semantic label map \(Y^*_r\) to estimate its accuracy \(\text{Acc} = \frac{TP}{TP + FN}\) (eliminating background label) and coverage \(\text{Cov} = \frac{TP}{TP + FN}\) (eliminating background label). (We manually label one view to calculate \(\text{Acc}\) and \(\text{Cov}\)) A larger \(\epsilon_H\) usually indicates low accuracy but high coverage and vice versa. To achieve a balance between accuracy and coverage, we estimate \(\epsilon_H\) by maximizing the following score function:

\[
\text{Score} = \text{Acc}^m \cdot \text{Cov} \quad (4)
\]

where \(m\) is a hyper-parameter indicating the importance of accuracy over coverage. A higher \(m\) tends to output more accurate semantic map. In this work, we choose \(m = 4\).

By applying the confidence-based thresholding, we can remove large amount of unreliable label predictions and improve the precision of the semantic maps in most of cases. In general, the filtered label maps are more accurate than the initial semantic maps that directly comes from the convolutional neural network. However, due to the filtering, each image may have some regions with missing label predictions. We can fill in semantic labels for these missing regions using stereo refinement.

### 4 Stereo-Based Semantic Refinement

For real-world scenes, PSPNet fails in multiple cases. When object B occludes object A, PSPNet may label the fragments neither label A or B. Further since PSPNet assumes a default minimal object size, small sized objects cannot be properly labeled either. Finally, generating a HCSM further decreases label coverage. The failure can cause serious visual artifacts. Our approach is to combine the HCSM with the disparity maps: an object, even fragmented by occlusions, generally has coherent depths. We employ such depth coherent to improve the semantic labeling quality as shown in Fig.6.

#### 4.1 Disparity Map Generation

There are a number of LF stereo matching techniques publicly available. However, nearly all of these approaches are tailored to handle LFs with ultra-small baselines such as the ones captured by the plenoptic camera. Since meaningful see-through effects require a wide synthetic aperture and inherently a large baseline between LF views, we resort to pair-wise stereo matching and then refine the results via warping.

Specifically, we employ a learning-based MC-CNN by Zbontar et al. [43] to first conduct pair-wise stereo matching and then refine the results via warping.

To estimate the parameter \(\epsilon_H\), we introduce a score function based on the quality of the label predictions satisfying the above condition. Denote the remaining pixel set as \(S_r\), we first use the semantic label map \(Y^*_r\) to estimate its accuracy \(\text{Acc} = \frac{TP}{TP + FN}\) (eliminating background label) and coverage \(\text{Cov} = \frac{TP}{TP + FN}\) (eliminating background label). (We manually label one view to calculate \(\text{Acc}\) and \(\text{Cov}\)) A larger \(\epsilon_H\) usually indicates low accuracy but high coverage and vice versa. To achieve a balance between accuracy and coverage, we estimate \(\epsilon_H\) by maximizing the following score function:

\[
\text{Score} = \text{Acc}^m \cdot \text{Cov} \quad (4)
\]

where \(m\) is a hyper-parameter indicating the importance of accuracy over coverage. A higher \(m\) tends to output more accurate semantic map. In this work, we choose \(m = 4\).

By applying the confidence-based thresholding, we can remove large amount of unreliable label predictions and improve the precision of the semantic maps in most of cases. In general, the filtered label maps are more accurate than the initial semantic maps that directly comes from the convolutional neural network. However, due to the filtering, each image may have some regions with missing label predictions. We can fill in semantic labels for these missing regions using stereo refinement.

### 4 Stereo-Based Semantic Refinement

For real-world scenes, PSPNet fails in multiple cases. When object B occludes object A, PSPNet may label the fragments neither label A or B. Further since PSPNet assumes a default minimal object size, small sized objects cannot be properly labeled either. Finally, generating a HCSM further decreases label coverage. The failure can cause serious visual artifacts. Our approach is to combine the HCSM with the disparity maps: an object, even fragmented by occlusions, generally has coherent depths. We employ such depth coherent to improve the semantic labeling quality as shown in Fig.6.

#### 4.1 Disparity Map Generation

There are a number of LF stereo matching techniques publicly available. However, nearly all of these approaches are tailored to handle LFs with ultra-small baselines such as the ones captured by the plenoptic camera. Since meaningful see-through effects require a wide synthetic aperture and inherently a large baseline between LF views, we resort to pair-wise stereo matching and then refine the results via warping.

Specifically, we employ a learning-based MC-CNN by Zbontar et al. [43] to first conduct pair-wise stereo matching and then refine the results via warping.

To estimate the parameter \(\epsilon_H\), we introduce a score function based on the quality of the label predictions satisfying the above condition. Denote the remaining pixel set as \(S_r\), we first use the semantic label map \(Y^*_r\) to estimate its accuracy \(\text{Acc} = \frac{TP}{TP + FN}\) (eliminating background label) and coverage \(\text{Cov} = \frac{TP}{TP + FN}\) (eliminating background label). (We manually label one view to calculate \(\text{Acc}\) and \(\text{Cov}\)) A larger \(\epsilon_H\) usually indicates low accuracy but high coverage and vice versa. To achieve a balance between accuracy and coverage, we estimate \(\epsilon_H\) by maximizing the following score function:

\[
\text{Score} = \text{Acc}^m \cdot \text{Cov} \quad (4)
\]

where \(m\) is a hyper-parameter indicating the importance of accuracy over coverage. A higher \(m\) tends to output more accurate semantic map. In this work, we choose \(m = 4\).

By applying the confidence-based thresholding, we can remove large amount of unreliable label predictions and improve the precision of the semantic maps in most of cases. In general, the filtered label maps are more accurate than the initial semantic maps that directly comes from the convolutional neural network. However, due to the filtering, each image may have some regions with missing label predictions. We can fill in semantic labels for these missing regions using stereo refinement.
Figure 3: Initial Semantic Labeling: We apply the Pyramid Scene Parsing (bottom row) to label individual LF views.

by minimizing the following energy function in two horizontal and two vertical directions:

\[
E(D) = \sum_p \{ C(p, D(p)) + \sum_{q \in N_p} P_1 \cdot 1(|D(p) - D(q)| = 1) + \sum_{q \in N_p} P_2 \cdot 1(|D(p) - D(q)| > 1) \}
\]

(6)

where \(C(\cdot)\) is the indicator function, \(P_1\) and \(P_2\) are smoothness penalties.

After sub-pixel enhancement, MC-CNN finally refines the disparity map by introducing a 5 × 5 median filter and the following bilateral filter:

\[
D_{Final}(p) = \frac{1}{M} \sum_{q \in N_p} D(q) \cdot g(\|p - q\|) \cdot 1(|D(p) - I(q)| < \epsilon_I)
\]

(7)

where \(g(\cdot)\) is the standard normal distribution, \(\epsilon_I\) is the intensity blurring threshold and \(M\) is the normalizing constant.

MC-CNN has shown superior performance over the state-of-the-art. However, directly using those disparities maps causes problems. This is mainly because the disparity map from a pair of images contains holes, especially on the left margin of the left image and the right margin of the right image.

Recall that an LF is composed of a number of rectified image pairs. This provides sufficient number of disparity maps to patch in the holes by warping and interpolating individual disparity map. Specifically, we generalize the interpolation scheme in [43] to multiple views.

Let \(D^R\) denote the reference disparity map in which we hope to patch the holes. Let \(\{D^{L1}, D^{L2}\}\) denote the left two disparity maps of the reference disparity map. An incorrect disparity pixel in \(D^R\) represents inconsistency between \(D^R\) and \(D^{L1}\). Therefore, we label each disparity \(d\) in \(D^R\) by performing the following consistency check on \(D^R\) and \(D^{L1}\):

\[
\text{label} = \begin{cases} 
\text{correct} & \text{if } |d - D^{L1}(p + d)| \leq 1 \text{ or } |d - D^{L1}(p + 2d)| \leq 1 \text{ for } d = D^R(p) \\
\text{mismatch} & \text{if } |d - D^{L1}(p + d)| \leq 1 \text{ for any other } d \\
\text{occlusion} & \text{otherwise}
\end{cases}
\]

For position \(p\) marked as occlusion, we perform a linear right search in \(D^{L1}\) until we find a correct position \(p'\) that satisfies \(p' - D^{L1}(p') = p\). If the search fails, we further search in \(D^{L2}\) until we find a correct position \(p''\) that satisfies \(p'' - D^{L2}(p'') = p\). For position marked as mismatch, we apply the same interpolation strategy as in [43]. We also consider the left margins in each view as occlusions and perform a linear search as above. The resulting disparity maps are illustrated in Fig. 4.

4.2 Disparity-Guided Semantic Refinement

To use disparity maps to refine semantic labeling, we assume that objects of different labels should have different depths. Notice if there are multiple objects lying at the same depth, they clearly will
not occlude each other and therefore our refinement will not affect the final result. Next, we fit a normal distribution $g_i(d)$ with respect to disparities of every label in $C$, where $d$ is disparity. It then becomes straightforward to label each pixel according to its disparity value, i.e., we assign label $i$ to pixel $[u,v,s,t]$ if $g_i(D(u,v,s,t)) > \epsilon_D$.

It is important to note that the assumption breaks down when there is a ground plane or two objects are close to each other. For example, in Fig. 5 a bicycle, a motorcycle and a horse are positioned on the same table where the table top plane maps to continuously varying disparity of a wide range. Consequently, different parts of the table will have the same disparity as the bicycle, the motorcycle, and the horse respectively. Brute-force application of the scheme above assigns the corresponding table parts as bicycle, motorcycle and horse.

To resolve this ambiguity, we employ a refinement process that incorporates both distance and normal cues, as shown in Fig. 5. For each conflict position $p$, we assume it belongs to either object A or B. We use two indicator functions $1_A(p)$ and $1_B(p)$ to represent that $p$ belongs to A or B. We use two measures to locally evaluate the possibility of each assignment for each conflict pixel. They comprise normal consistency and distance consistency:

$$E = p_n \cdot E_n(1_A(p), 1_B(p)) + p_d \cdot E_d(1_A(p), 1_B(p))$$

$E_n$ represents the normal consistency and $E_d$ represents the distance consistency.

To measure normal consistency, we extract confident regions $R^{(A,B)}$ of A and B from HCSM, and calculate normals $n$ for each pixels in $R^{(A,B)}$ from the disparity map. Then calculate the normal consistency as:

$$C^{(A,B)} = \frac{\sum_{r \in R^{(A,B)}} n(r) \cdot n(r)}{\sum_{r \in R^{(A,B)}} \| n(r) \|}$$

where $\cdot$ represents inner product, $m$ if the normal of $p$ from the disparity map, $C^{(A,B)}$ will be close to 1 when the normal of $p$ is consistent with $R^{(A,B)}$, i.e., parallel with normals in $R^{(A,B)}$. This indicates that $p$ probably belongs to A (B). Otherwise, $p$ is less likely to belong to A (B). Thus, $E_n$ in Eq.8 is:

$$E_n = C_A \cdot 1_A(p) + C_B \cdot 1_B(p)$$

We also calculate the distance consistency as follows:

$$E_d^{(A,B)} = \frac{\min_{r \in R^{(A,B)}} \| p - r \|^2}{\max_{r \in R^{(A,B)}} \| p - r \|^2} \cdot 1_A(p) + \frac{\min_{r \in R^{(A,B)}} \| p - r \|^2}{\max_{r \in R^{(A,B)}} \| p - r \|^2} \cdot 1_B(p)$$

$E_d$ will be close to 0 when $p$ is close to the region of A. Otherwise, it will be a large value.

We consider normal consistency only when one of A or B is ground region. In this case, we can easily calculate the normal of the ground region from the disparity map.

5 SEMANTIC SEE-THROUGH RENDERING

Once we manage to assign a label and disparity to each ray in the light field, we can conduct semantic see-through (SST) rendering. We first modify the traditional LF rendering by adding a new weighting function:

$$W^*(s_r, d_r, d_f) = \max(0, -\frac{(d_f - D_{\text{min}}^f)(d_f - D_{\text{max}}^f)}{(D_{\text{max}}^f - D_{\text{min}}^f)/2)^2})$$

where $D_{\text{min}}^f = \min(d_r : s_r = s_{r_{\text{max}}})$ and $D_{\text{max}}^f = \max(d_r : s_r = s_{r_{\text{max}}})$ define the depth range of all rays that has the same label with $r_{\text{max}}$. If $s_r$ is far from $s_{r_{\text{max}}}$ then $W^*$ becomes 0. We also calculate the distance consistency as follows:

$$L(s,t,s + \frac{u' - s}{d_f}, t + \frac{v' - t}{d_f}) = A(s,t)$$

$$W(s_r, d_r, d_f)$$ is the (normalized) weighting function of both the semantic label $s_r$ and the depth $d_r$ of ray $r$. An example of out weighting scheme is shown in Fig.7.

**Depth-guided Weighting.** We design the special weighting function in order to achieve better see-through results (Fig.7). Suppose a desired ray $r_d$ intersects the focal plane at point $p_d$. For each sub-aperture view $C(s,t)$, we trace the point back to $C$ as ray $r_{st}$. Since we know the corresponding disparity map for $C(s,t)$, we obtain two depth values, $d_f$ as the focal plane depth and $d_{r_{st}}$ as the depth/disparity of the ray. Conceptually if the ray’s depth coincides with the focal plane, the ray should be in-focus, and hence has a higher weight. Therefore, we assign a Gaussian function as its weight:

$$W^*(d_{r_{st}},d_f) = e^{-\frac{(d_f - d_{r_{st}})^2}{2\sigma_d^2}}$$

where $\sigma_d$ is the standard deviation and it determines how many rays are impactful, i.e., may has bigger weights. As $\sigma_d$ approaches infinity, $W^*(d_{r_{st}},d_f)$ will be 1 and the algorithm degrades to traditional refocusing algorithm.

However, this weighting function can not satisfy our rendering demand because it will assign a extremely small weight to some objects are almost invisible. To overcome this problem, we introduce a constant minimum weight threshold $C_1$ and the depth weighting function becomes:

$$W(d_{r_{st}},d_f) = (1 - C_1)W^*(d_{r_{st}},d_f) + C_1$$

$C_1$ determines a fading factor. As $C_1$ decreases, objects away from the focal depth get more impactful. By setting different values to $C_1$, we can achieve different see-through effects.

**Semantic-guided Weighting.** Consider refocusing on an object under occlusions. The first issue is when an object’s scale is much smaller than the impactful depth range defined by $\sigma_d$. In this situation, the renderer will still assign considerable weights to its obstructions. As a result, the object will still appear with ghosting effects. The second issue is when two objects are overlapping. Consider a scene where a panda is climbing a bamboo tree, from the camera’s perspective, they have similar depths. Therefore, depth-guided weighting scheme will not distinguish them very well and produce expected occlusion-removed results.

To overcome the first problem, we apply a semantic weighting function (Fig.7). We assume each object’s weight follows the quadratic distribution and define semantic weight function as following:

$$W^*(s_r, d_f) = \max(0, -\frac{(d_f - D_{\text{min}}^f)(d_f - D_{\text{max}}^f)}{(D_{\text{max}}^f - D_{\text{min}}^f)/2)^2})$$

where $D_{\text{min}}^s = \min(d_r : s_r = s_{r_{\text{max}}})$ and $D_{\text{max}}^s = \max(d_r : s_r = s_{r_{\text{max}}})$ define the depth range of all rays that has the same label with $s_{r_{\text{max}}}$. If $s_r$ is far from $s_{r_{\text{max}}}$ then $W^*$ becomes 0.
Figure 5: Depth-guided Semantic Label Refinement. Given the initial HCSM, we first model, for each label, the distribution of its disparity (c). For each unlabeled (low confidence) pixel, we query its disparity (d) into (c) to determine its optimal label. Finally, we remove the outliers caused by the ground plane via normal comparison (f).

![Figure 5: Depth-guided Semantic Label Refinement](image)

Same as depth weight blending, we define a minimum weight threshold $C_2$ so that rays do not get dark when the focal plane depth is out of their depth range. The semantic weighting function then becomes

$$W(s_{rt}, d_f) = (1 - C_2)W^*(s_{rt}, d_f) + C_2$$  \hspace{1cm} (16)

$C_2$ determines another fading factor.

Figure 6: Semantic labeling before and after stereo refinement. Left shows the input view; Middle shows the high confidence semantic map (HCSM); Right shows the stereo refined results.

![Figure 6: Semantic labeling before and after stereo refinement](image)

Figure 7: SST Rendering: We assign every ray a weight in terms of its disparity and semantic label. The three rows show how weighting changes in terms of the focal depth.

![Figure 7: SST Rendering](image)

In the second scenario where two objects occlude each other, it is non-trivial to tell which object is more important and should be clear. Our implementation utilizes a user interface to choose the desired one manually. We then reduce the weight of other objects by a factor of 2.
By multiplying $W(d_{r\theta}, d_f)$ and $W(s_{r\theta}, d_f)$, we obtain a joint weight function:

$$W(s_r, d_r, d_f) = W(d_{r\theta}, d_f) \cdot W(s_{r\theta}, d_f)$$  \hspace{1cm} (17)

Normalizing over $st$ gives us the final

$$W_{norm}(s_r, d_r, d_f) = \frac{W(d_{r\theta}, d_f) \cdot W(s_{r\theta}, d_f)}{\int W(d_{r\theta}, d_f) \cdot W(s_{r\theta}, d_f)dsdt}$$  \hspace{1cm} (18)

The complete algorithm can be easily implemented on the GPU as in Algorithm 1.

**Algorithm 1** Real-time semantic light field rendering

Require: Light field $C(s, t)$; Depth map $D(s, t)$; Semantic map $S(s, t)$

for all pixel $(x, y)$ on desired image plane do

1. $p :=$ the pixel under consideration
2. $q :=$ the intersection of $\overrightarrow{O_p}$ with focal plane
3. $W_{sum} := 0$
4. $c(x, y) := BLACK$
5. for all reference camera $(s, t)$ in aperture do
6. $r :=$ the ray through $q$ and $O(s, t)$
7. $(u, v) :=$ projection of $q$ onto the image plane of $(s, t)$
8. Compute $W_{depth}$ using Eq.14
9. Compute $W_{semantic}$ using Eq.16
10. $W := W_{depth} \cdot W_{semantic}$
11. $W_{sum} := W_{sum} + W$
12. $c(x, y) := c(x, y) + W \ast S(s, t, u, v)$
13. end for
14. $c(x, y) := c(x, y) / W_{sum}$
15. end for

6 EXPERIMENTAL RESULTS

To evaluate our algorithm, we set up different kinds of scenes to capture real data. The supplementary materials are also provided to show more results. We first build a mechanical camera array using a Canon 760D camera to record indoor light field data in $1920 \times 1280$ resolution for each subaperture view. We mount the camera on a motor to control its position precisely. We also build a $6 \times 6$ GoPro camera array to capture outdoor scenes. The resolution of each subaperture view is $2000 \times 1500$ and all cameras can be synchronized by the official synchronizer (Fig.1). Moreover, we test our rendering scheme on Stanford light field archive [34]. For our captured scenes, the input are only light field images. All our experiments are conducted on a computer with Intel Xeon CPU E5-1620 v4 @ 3.5GHz, 16GB memory and Titan X graphics card.

To conduct our SST algorithm, we need to compute the semantic labels and depth maps for each subaperture view first. We implement our stereo-based semantic refinement algorithm in MATLAB and compare our results with that of PSPNet – one of the most advanced semantic segmentation approaches. Fig.6 shows that our results exhibit much higher accuracy and sharper boundaries. Our refinement can handle heavily occluded areas which are very hard to deal with in single image semantic segmentation. As for the depth map generation, we compare our approach with MC-CNN method. Due to the occlusion existing, MC-CNN method generates large black holes especially at the boundaries of the objects as shown in the closeups of Fig.4. Instead, our approach patches mismatched and occluded positions effectively (Fig.4) by exploiting disparity consistency among adjacent views. The improvement of disparity maps also in turn enhances our semantic segmentation refinement.

We implement SST on a NVIDIA GPU. Fig.8 shows our SST rendering results on the “transportation” light field dataset captured by our Canon device. When the focus changes from the foreground to
the background, different models (the toy horse, the bicycle model and the motorcycle model) become clear. Even the motorcycle is occluded by the bicycle, our SST can still render the complete motorcycle thanks to the additional semantics and depth information. Our SST method works on not only the large scale scenes, but also the micro scenes like cells. Fig.10 shows our SST rendering results on the “cell” light field. We change the focal plane to distinguish organelles at two depth layers. The bushy leaves and the toys. Our rendering scheme produces a remarkable see-through effects while this is impossible by using traditional rendering scheme as shown in Fig.9.

We also compare the rendering results of SST with traditional refocusing’s on both indoor and outdoor scenes as shown in Fig.12. SST remarkably reduces ghosting effects and produces a more clear view of occluded objects than traditional refocusing. In the first scene, we capture the “bush” light field. A man sits on a sofa behind the heavily foliaged bush which extremely occludes him. Traditional refocusing generates obvious ghosting artifacts on the face and body, because the method doesn’t take the label information of the environment into account. Instead, SST leverages the depth and labels to assign each ray with different weight and therefore remove nearly all foreground bush when the focus is on the man. Seeing through the occluders is crucial in security surveillance. The second scene is the ‘pedestrian’ light field. A man walks in front of a white car. When focusing on the car, SST decreases the impacts of the pedestrian, and renders the car more clearly. When focusing on the building at behind, traditional scheme renders a mixed color on the building surface, while SST renders the correct color. To test the robustness of our algorithm, we also compare our SST with regular refocusing method on the Stanford ‘CD cases and poster’ light field (the bottom row in Fig.12). Our SST still achieves superb see-through effects when the focus is on the CD cases and poster individually.

Although our SST has achieve unprecedented see-through rendering results, it still has some drawbacks (Fig.11). Our SST algorithm is based on input disparity maps and semantic labeling, so the quality of the input is crucial to generate good results. SST can fail when stereo matching algorithm does not compute reasonable disparity values, and PSPNet incorrectly label regions of objects at the same time. Because in this case, we are not able to refine semantic maps and SST is unable to differentiate rays according to its label and depth.

7 CONCLUSIONS AND FUTURE WORK

We have presented a novel semantic see-through (SST) rendering technique that can provide high quality refocusing effects. Different from traditional refocusing, our technique effectively exploits semantic ray labeling and learning-based stereo matching. By designing proper blending functions, our technique manages to mitigate visual artifacts caused by foreground residues and at the same time maintains smooth spatial, angular and focal transitions.

To our knowledge, this is the first light field rendering technique that explicitly incorporates semantic analysis. There is also an emerging trend on integrating semantic labeling into 3D reconstruction [16, 20] such as stereo matching and volumetric reconstructions. Our immediate task is to explore visibility analysis in these techniques for more general image-based rendering techniques such as view morphing [29], lumigraph [10], and surface light field rendering [41]. Further, more sophisticated label sets, e.g., to separate various materials, can be used in our framework. This can potentially benefit 3D reconstruction highly view-dependent materials. For example, we intend to explore coupling semantic analysis with specular light field reconstruction [35], light field super-resolution [24], and hyper-spectral light field imaging [13].
Figure 12: SST Rendering vs. Regular Refocusing on the "bush" light field (top row), "pedestrian" light field (mid row), and the Stanford "CD cases and poster" light field (bottom row). From left to right, we change the focal plane. Our technique achieves superb see-through capability while maintaining smooth focus transitions. Please refer to the supplementary video for the complete focal sweep results.
