Abstract

Many 4D light field processing methods and applications rely on superpixel segmentation, for which occlusion-aware view consistency is important. Yet, existing methods often enforce consistency by propagating clusters from a central view only, which can lead to inconsistent superpixels for non-central views. Our proposed approach combines an occlusion-aware angular segmentation in horizontal and vertical epipolar plane image (EPI) spaces with a clustering and propagation step across all views. Qualitative video demonstrations show that this helps to remove flickering and inconsistent boundary shapes versus the state-of-the-art light field superpixel approach (LFSP [25]), and quantitative metrics reflect these findings with greater self similarity and fewer numbers of labels per view-dependent pixel.

1. Introduction

Superpixel segmentation attempts to simplify a 2D image into small regions to lessen future computation, e.g., for later graph inference in interactive object selection. Desirable superpixel qualities vary between applications [18], but generally we wish for them to be accurate, i.e., to adhere to image edges; to otherwise be compact in shape, and to be efficient to compute (see Stutz et al. for a review [17]).

Light fields represent small view changes onto a scene, e.g., an array of $9 \times 9$ 2D image views (‘4D’). Processing light fields is computationally harder due to the increased number of pixels, but many of these pixels are similar because the view change is small. As such, we have much to gain from simplifying light field images into superpixels. This introduces a new desirable property for our light field superpixels: we wish them to be view consistent, e.g., they do not drift, swim, or flicker as the view changes, and we wish superpixels to include all similar pixels across views such that they respect occlusions. This is particularly important for applications which will use every light field view, such as editing a light field photograph for output to a light field display.

It is difficult to achieve the four properties of accuracy, compactness, efficiency, and view consistency. Existing approaches often propagate superpixel labels into other views via a central-view disparity map. However, this can cause inconsistency for regions occluded in the central view, e.g., the recent light field superpixel (LFSP) method [25] does not always maintain view consistency.

We can attempt to estimate per-view disparity maps, but this can be difficult for small occluded regions in off-central views.

We propose a method for accurate and view-consistent superpixel segmentation on 4D light fields which implicitly computes disparity per view and explicitly handles occlusion (Fig. 2). First, we robustly segment horizontal and vertical epipolar plane images (EPIs) of the 4D light field. This provides view consistency in an occlusion-aware way by explicit line estimation, depth ordering, and bipartite graph matching. Then, we combine the angular segmentations in horizontal and vertical EPIs via a view-consistent clustering step. Qualitative results (Fig. 1) show that this reduces flickering from inconsistent boundary shapes when compared to the state-of-the-art LFSP approach [25], and quantitative metrics reflect these findings with improved view consistency scores.

Given a disparity map for the central view, Zhu et al. [25] posed the oversegmentation problem in a variational framework, and solved it efficiently using the Block Coordinate Descent algorithm. While their method generates compact superpixels, these sometimes flicker as shape changes across views (Fig. 1). Our approach specifically enforces view consistency, which is desirable for many light field applications.

### 3. View-consistent Superpixel Segmentation

#### Definitions

Given a 4D light field $LF(x,y,u,v)$, we define the central horizontal row of views $H = LF(x,y,u,v)$ and central vertical column of views $V = LF(x,y,u,v)$. Each view $I \in H$ contains a set of EPIs $E_i(x,y,u,v)$, with corresponding $I \in V$ containing $E_j(y,x,v,u) = I(x,y,u,v)$.

With a Lambertian reflectance assumption, a 3D scene point $x \in \mathbb{R}^3$ is a 4D vector $x \in \mathbb{R}^4$.

Our algorithm has three major steps (Fig. 2).

**Step 1: Line Detection (Sec. 3.1):** Providing view-consistent and occlusion-aware segmentation relies critically on accurate edge line detection (i.e., disparity estimation at edges). As such, we begin by creating two slices of the light field as EPIs, one each for the central horizontal and vertical directions. Then, we robustly fit lines with the specific goal of later handling occlusion cases.

**Step 2: Occlusion-aware EPI Segmentation (Sec. 3.2):** Next, we must reason about the scene order of detected lines to pair them into segments. This is solved via a bipartite graph matching process, which allows us to strictly enforce occlusion awareness. It produces per-EPI view-consistent regions in horizontal and vertical dimensions, which must be merged spatially.

**Step 3: Spatio-angular Segmentation via Clustering (Sec. 3.3):** Finally, we merge EPI regions into a consistent segmentation via a segment clustering, which uses our estimated disparity to regularize the process. Remaining unlaabeled off-central-directions occluded pixels are labeled via a simple propagation step.
3.1. Line Detection

For robust occlusion handing, we must accurately detect the intersections of lines in EPIs (Fig. 3). However, classical edge detectors like Canny [3] and Compass [15] often generate curved or noisy responses at line intersections, which makes later line fitting and occlusion localization difficult. Instead, we propose an EPI-specific method. Note: We describe line detection for the central horizontal views; central vertical views follow similarly.

EPI Edge Detection We take all EPIs $E_i(x, u)$ (size $w \times h$) from the horizontal central view images $I \in \mathcal{H}$. We convolve them with a set of 60 oriented Prewitt edge filters with each representing a particular disparity. We filter only the central views for efficiency, and later on will propagate their edges across all light field views. To detect small occluded lines, we use $2h \times 2h$ filters and convolve the entire $(x, u)$ space. This effectively extends occluded edge response to span the height of the EPI.

From this, we pick the filter with maximal response per pixel, which is a disparity map $Z$ at edges, and we take the value of the filter response as an edge confidence map $C$. Then, we perform non-maximal suppression per EPI. To suppress false responses in regions of uniform color, we modulate edge response by the standard deviation of a $3 \times 3$ window around each pixel in the original EPI [11]. Our final $C$ map has clean intersections (Fig. 3). Algorithm 1 summarizes our approach.

Line Fitting To create a parametric line set $L$, we form lines $l_i$ from each pixel in $C$ in confidence order, with line slopes from $Z$. As we add lines, any pixels in $C$ which lie within an $\lambda$-pixel perpendicular distance of the line $l_i$ are discarded. $\lambda$ determines the minimum feature size that our algorithm can detect. In all our experiments, we set $\lambda = 0.2h$. We proceed until we have considered all pixels in $C$. For efficiency, we detect edges and form line sets in a parallel computation per EPI.

**Algorithm 1:** EPI edge detection

```plaintext
FindEdgesEPI (E, F)
Input: E: A $w \times h$ EPI
F: A set of 60 $2h \times 2h$ directional filters.
Output: An edge slope map $Z$ with confidences $C$.

foreach $f_i \in F$ do
  $r_i \leftarrow E \circ f_i$;
end

foreach pixel location $(u, v) \in I$ do
  $Z(u, v) \leftarrow \text{argmax}_i r_i(u, v)$;
  $C(u, v) \leftarrow \max_i r_i(u, v)$;
  $V(u, v) \leftarrow \text{StdDev}(I(N(u, v)))$ for neighborhood $N(u, v)$ around $(u, v)$;
end
$C \leftarrow \text{NonMaxSuppress}(C) \circ V$;
return $Z, C$
```

![Figure 3](image)

**Figure 3:** Our method can detect edge intersections more accurately than the Canny or Compass methods. These intersections provide valuable occlusion information.

**Outlier Rejection** We wish to exploit information from across the spatio-angular light field. As such, we defer outlier rejection until after we have discovered $L$ for each EPI in each horizontal view, and then project all discovered lines into the central view. Given this, we wish to keep both (a) high confidence lines, and (b) low confidence lines which have similar spatio-angular neighbors, and reject faint lines caused by noise.

Given a line $l_i \in L$ with confidence $c_i$ and disparity $z_i$, we count the number of lines within a $p \times q$ pixel spatial neighborhood $N(l_i)$, and weight this number by the confidence $c_i$:

$$A(l_i) = \{l_k \in N(l_i) \mid z_k = z_i\}. \quad (1)$$

Then, we discard a line $l_i \in L$ if:

$$\frac{c_i |A(l_i)|}{pq} < \tau, \quad (2)$$

where $p$ and $q$ are 1/15th of the width and height of the light field, and $\tau = 8 \times 10^{-5}$. This is similar to Canny’s use of a double threshold to robustly estimate strong edges: strong lines must have a confidence greater than $\tau pq$, and weak lines must have $\tau pq/c_i$ neighbors at the same disparity.

**Spatial Multi-scale Processing** To detect broader lines and improve consistency between neighboring EPIs, we compute coarse-to-fine edge confidence across a multi-scale pyramid with $2 \times$ scaling in the spatial dimensions only. At each scale and after the outlier removal processes, we double the $x$ location of detected lines intersects, and repeat each line twice along $u$. We replace any lines in a coarser scale which are close to lines in a finer scale. That is, we replace a coarse line only if both of its end points are within $\lambda$ pixels of the fine line. Thus, broader spatial lines which are not detected at a finer scale are still kept.

With this, we have now discovered a line set $L$ for each EPI of the central horizontal and vertical views of our light field.
The occlusion direction is given by the side of the foreground line determined by the relative slope of the two lines. This direction can be found by considering a small region of the edge image around the point of intersection. The side of the foreground line on which the background line is visible defines the direction of occlusion.

Occluding lines can only match with other lines in the matching region and determine the correct matching (Fig. 4c). However, it cannot match with any line that lies beyond the foreground line on which the background line is visible. The foreground line on which the background line is visible defines an under-constrained problem in which segment order cannot be uniquely determined.

We solve it by considering a small region around the point of intersection in the edge image $E$, which allows us to constrain the occlusion direction and determine the correct matching (Fig. 4b). The occlusion direction is given by the side of the foreground line in which the background line is visible. The foreground line is determined by the relative slope of the two lines.

The sequence of steps to narrow down the potential matches for each line is shown in Figure 4. Once we have omitted any line pairings which violate the occlusion order, we pose line matching as a two-step maximum value bipartite matching problem on a complete bipartite graph $G(L,L,E)$ and solve it using Dulmage-Mendelson decomposition. In the first step, we match only intersecting lines to resolve occlusions. In the second step, all remaining lines are matched. We compute line distance as:

\[
\text{Distance}(l_i,l_k) = (\omega_d|t_i - t_k - b_i + b_k| + (1 - \omega_d)|t_i + b_i - t_k - b_k|)^{-1},
\]

where $t_i$ and $b_i$ are the line intercepts $l_i$ at the top and bottom of the EPI image. $\omega_d$ is a constant which determines the relative importance of disparity similarity over spatial proximity of lines.

Finally, to prevent forming large superpixels in uniform regions, we recursively split any segment that has a width larger than 15 pixels by adding new lines. To regularize segments across the vertical and horizontal EPI directions—especially in textureless regions—the slope of new lines is always set to match the disparity of the vertical segment covering that spatial region.

The procedure is given in Algorithm 2. Figure 5 shows an example EPI result after the computations of Sections 3.1 and 3.2.

---

**Algorithm 2: EPI line segment matching.**

**SegmentEPI** ($L$)

**Input:** $L$: An ordered list of line segments bounded by the top and bottom edges of EPI $I$.

**Output:** A set $M \in L \times L$ of line couplings.

Create the complete bipartite graph $G = (L,L,E_f)$ for matching all occluding lines:

$S \leftarrow \text{OccludingLines}(L)$;

**foreach** $l = (l_i,l_j) \in E_f$ **do**

**if** $l_i \notin S$ **and** $l_j \notin S$ **then**

$w(e) \leftarrow -\infty$;

**else if** $l_j$ does not lie to the left of $l_i$ **then**

$w(e) \leftarrow -\infty$;

**else if** $\exists k \in S$ to the left of $l_i$

Distance($l_i,k$) $< \text{Distance}(l_i,l_j)$ **then**

$w(e) \leftarrow -\infty$;

**else**

$w(e) \leftarrow \text{Distance}(l_i,l_j)$;

**end**

$A \leftarrow \text{MaxBipartiteMatching}(G)$;

$U \leftarrow \{l \in L \mid (\exists k)[k \in L \land (l,k) \in A]\}$;

$V \leftarrow \{k \in L \mid (\exists l)[l \in L \land (l,k) \in A]\}$;

Create the complete bipartite graph $H = (L \setminus U, L \setminus V, E)$ for matching all other lines;

**foreach** $e = (l_j,l_k) \in E$ **do**

**if** $l_k$ does not lie to the left of $l_j$ **then**

$w(e) \leftarrow -\infty$;

**else**

$w(e) \leftarrow \text{Distance}(l_j,l_k)$;

**end**

$B \leftarrow \text{MaxBipartiteMatching}(H)$;

**return** $A \cup B$.
3.3. Spatio-angular Segmentation via Clustering

Our occlusion-aware segmentation per EPI must now be combined across different EPIs as, currently, we have no correspondence between the horizontal and vertical EPI segments (other than the large-region split lines added in the previous step). We address this by jointly clustering the segments in the central view of the light field using k-means in \((x,y,d,L^*,a^*,b^*)\) space (Fig. 6).

This clustering approach with disparity \(d\) might seem similar to methods which exploit a central depth map for propagation, like LFSP [25]. However, our method is view consistent: our EPI segment-based computation allows us to estimate \(d\) for every light field view, including those segments occluded from the central view. These are all considered within the clustering.

For each segment, we compute the average pixel value in the CIELAB color space: \(L^*,a^*,b^*\). We define the disparity \(d\) from the larger (deeper) slope of the two segment lines. For segments in horizontal EPIs, \(y\) equals the EPI index and we determine \(x\) to be the midpoint of the segment lines in the central view. For vertical EPIs, we reverse this relation. The number of clusters is user specified and determines superpixel size. We seed clusters at uniformly-distributed spatial locations [1], and assign \(x,y,d,L^*,a^*,b^*\) from the segment center closest in image space.

Within the feature vector, \(x,y\) have weight 1 and \(L^*,a^*,b^*\) have weight 3. We normalize \(d\) given our current scene estimates then weight it by 120. This larger weight helps the method not to cluster across occlusions, which usually have different disparities.

Clustering within the central view allows us to correspond and jointly label the horizontal and vertical EPI segments, and to provide spatial coherence. However, the boundaries from these two EPI segmentations do not always align. Thus, after projecting these segments into all light field views, we discard labels for pixels where the two segmentations disagree.

3.3.1 Label Propagation

At this point, our only unlabeled pixels are those either occluded from or in disagreement between both central sets of views in the vertical and horizontal directions. We note that 1) the set \(U\) of unlabeled pixels is sparse even within a local neighborhood; and that 2) at this stage, we know the disparity of each labeled pixel in the light field. As such, we minimize a cost with color, spatial, and disparity terms to label the remaining pixels.

Given an unlabeled pixel \((x,y)\in U\) in light field view \(I_{u,v}\), let \(\mathcal{L}(x,y)\) define the set of labeled pixels in a spatial neighborhood around \((x,y)\). For every pixel \((p,q)\in \mathcal{L}(x,y)\), let \(\ell(p,q)\) denote its label, and \(d(p,q)\) its disparity. Moreover, let \(I_{s,r}(\cdot,\cdot)\) represent the color of any pixel, labeled or unlabeled, in light field view \(I_{s,r}\). We define the cost of assigning \((x,y)\) label \(\ell(p,q)\) as:

\[
E_{(x,y)}(\ell(p,q)) = \omega_c(I_{u,v}(x,y) - I_{u,v}(p,q))^2 + \omega_s((x-p)^2 + (y-q)^2)
+ \omega_d \left( \sum_s \sum_r I_{u,v}(x,y) - I_{s,r}(x+d(p,q),y+d(p,q)) \right).
\]

We set weights empirically: \(w_c = 1, w_s = 1, w_d = 1e^{-5}\).

Label assignment total cost is \(E = \sum_{(x,y)\in U} E_{(x,y)}(\cdot)\). We efficiently compute this by minimizing \(E_{(x,y)}(\cdot)\) per pixel. Along with finding \(\ell(x,y)\), we set \(d(x,y)\) equal to \(d(\text{argmin}_{\ell(x,y)}E_{(x,y)})\), which allows us to project newly-assigned labels to any unlabeled pixels in other views. In practice, this strategy only requires minimization over the central row and column of light field views, with the few remaining pixels in off-center views after projection labeled by nearest neighbor assignment.

4. Experiments

4.1. Setting

Datasets We use synthetic light fields with both ground truth disparity maps and semantic segmentation maps. From the HCI Light Field Benchmark Dataset [22], we use the four scenes with ground truth: papillon, buddha, horses, and still life. Each light field image has \(9\times9\) views of \(768\times768\) pixels, except horses with \(1024\times576\) pixels. For real-world scenes, we use the EPFL MMSPG Light-Field Image Dataset [26]. These images were captured with a Lytro Illum camera (15\times15 at 434\times625). Please refer to our supplementary materials for more results.

Baselines We compare to the state-of-the-art LFSP (light field superpixel segmentation) approach of Zhu et al. [25]. This method takes as input a disparity map for the central light field view. We apply their method on the disparity estimates from Wang et al. [20, 21] as originally used in the Zhu et al. paper, and on ground truth disparity. Comparing these two results shows the errors which are introduced from inaccurate disparity estimation.

We also compute a k-means clustering baseline, which is similar in spirit to RGBD superpixel methods like DASP [24] methods. Given a disparity map for the central light field view, we convert the input images to CIELAB color space and form a
vector \( f = (x, y, d, L^*, a^*, b^*) \) for each pixel in the central view of the light field. Then, from uniformly-distributed seed locations, we cluster using the desired number of output superpixels, and project these labels into other views. For any pixels in non-central views which remain unlabelled, we assign the label of the nearest neighbor based on \( f = (x, y, L^*, a^*, b^*) \). For each feature, we use the same weight parameters as in our method. As for LFSP, we compute results using ground truth disparity maps and with the estimation method of Wang et al. [20, 21].

**4.2. Metrics**

We use two view-consistency-specific metrics: self similarity error [25] and number of labels per pixel; explained below. We also use three familiar 2D boundary metrics: achievable accuracy, boundary recall, and undersegmentation error; we explain these in our supplemental material. Achievable accuracy, self similarity, and number of labels per pixel describe overall accuracy and consistency across views. Boundary recall and undersegmentation error describe characteristics of over segmentation [14]. As a measure of superpixel shape, we use the compactness metric from Schick et al. [16]. We compute each metric across average superpixel sizes of 15–40 square (225–1600 pixels each).

**Self Similarity Error** As defined in Zhu et al. [25], we project the center of superpixels from each view into the central view, and compute the average deviation versus ground truth disparity. Smaller errors indicate better consistency across views.

**Number of Labels Per View-dependent Pixel** We compute the mean number of labels per pixel in the central view as projected into all other views via the ground truth disparity map. This gives a sense of the number of inconsistent views on average (cf. HCI dataset with 81 input views). For ease of computation, we discard pixels which are occluded in the central view.

**4.3. Results**

Figure 7 shows all metrics averaged over all four scenes; our supplementary material includes per-scene metrics. For qualitative results, please see our supplemental video.

**View Consistency** Our method outperforms both LFSP and the k-means baselines using estimated disparity maps (Fig. 7(a)). These findings are reflected in qualitative evaluation where we reduce view inconsistencies such as flickering from superpixel shape change over views (Fig. 8). Using ground truth disparity maps, our method outperforms LFPS on both metrics, but only outperforms k-means on self similarity error: k-means with ground truth disparity produces fewer numbers of labels per pixel than our method. As a reference for interpretation, the small baselines cause occlusion in \(~3–5\%\) of light field pixels.

**Achievable Accuracy, Boundary Recall, and Undersegmentation Error** Our method outperforms LFSP for all three metrics on both estimated and ground truth disparity for all superpixel sizes (Fig. 7(c)). For smaller superpixel sizes (15–25), we are competitive in accuracy and undersegmentation error with k-means using ground truth disparity; at larger sizes k-means is better. Our method recalls fewer boundaries than k-means: we occasionally miss an edge section during step 1, which defers these regions to our less robust final propagation step for unlabeled pixels instead. However, k-means can create very small regions (Fig. 8) which are broadly undesirable.

**Compactness** Our method is competitive with LFSP at smaller superpixel sizes (15–25), and better at larger sizes (Fig. 7(b)). The k-means baseline generates the least compact superpixels of the tested methods, even with ground truth disparity. As we just saw, this shape freedom helps it recall more boundaries.

**Computation Time** We use an Intel i7-5930 6-core CPU and MATLAB for our implementation. We report times on the 9 × 9 view light fields with images of 768 × 768 pixels. Disparity map computation for Wang et al. takes \(~8\) minutes, which is a pre-process to both the k-means baseline and LFSP. LFSP itself takes \(~2\) minutes, with k-means taking \(~2.5\) minutes. Our approach implicitly computes a disparity map and takes \(~3.3\) minutes total.

**5. Discussion and Limitations**

Our approach attempts to compute a view-consistent superpixel segmentation and produces competitive results; however,
some issues still remain as not every pixel in the light field is view consistent. First, our occlusion-aware EPI segmentation is explicitly enforced by matching rules; however, the clustering step in Section 3.3 does not explicitly handle occlusion—this is only softly considered within the clustering by a high disparity weight. Further, for efficiency, we rely on only the central horizontal and vertical views. When segment boundary estimates do not align between these two sources, or when pixels are occluded from both of these sets of views, we rely on our less robust label propagation (Section 3.3.1) which is not occlusion aware and uses no explicit spatial smoothing, e.g., via a more expensive pairwise optimization scheme. Both of these issues can cause minor label ‘speckling’ at superpixel boundaries. We hope to improve these aspects of our method in future work.

While a valued resource for its labels, the HCI dataset \[22\] has minor artifacts in its ground truth disparity, such as jagged artifacts on the wooden plank in the ‘buddha’ scene. It is no longer supported and a replacement exists \[8\]; however, this does not include object segmentation labels for non-central views, which makes evaluating view consistency with it difficult.

Our Lambertian assumption makes it difficult to handle specular objects: view-dependent effects break the assumption that a 3D scene point maps to a line in EPI space, e.g., in the HCI dataset ‘horses’ scene where all methods have trouble. Further, as the normalized ratio of area to perimeter, compactness is only a measure of average shape across the superpixel, and sometimes our superpixel boundaries have higher curvature than LFSP.

6. Conclusion

We present a view-consistent 4D light field superpixel segmentation method. It proceeds with an occlusion-aware EPI segmentation method which provides view consistency by explicit line estimation, depth ordering constraints, and bipartite graph matching. Then, we cluster and propagate labels to produce per-pixel 4D labels. The method outperforms the LFSP method on view consistency and boundary accuracy metrics even when LFSP is provided ground truth disparity maps, yet still provides similar shape compactness. Our method also outperforms a depth-based k-means clustering baseline on view consistency and compactness metrics, and is competitive in boundary accuracy measures. Our qualitative results in supplemental video show the overall benefits of view consistency for light field superpixel segmentation.

Acknowledgements

We thank Kai Wang for proofreading, and a Brown OVPR Salomon Faculty Research Award and an NVIDIA GPU Award for funding this research. Min H. Kim acknowledges support by Korea NRF grants (2019R1A2C3007229, 2013M3A6A6073718) and Cross-Ministry Giga KOREA Project (GK17P0200).
Figure 8: Superpixel segmentation boundaries and view consistency for the k-means baseline, LFSP [25], and our method. Disparity maps for LFSP and $k$-means were calculated using the algorithm of Wang et al. [20, 21]. Top two rows: HCI dataset [22]; we highlight superpixels which either change shape or vanish completely across views. Bottom two rows: EPFL Lytro dataset [26]. Our superpixels tend to remain more consistent over view space, which can be easily seen as reduced flickering in our supplementary video. Note: Small solid white/black regions appear when superpixels are enveloped by the boundary rendering width. $k$-means tends to have more of these regions which helps it increase boundary recall, but this behavior is not useful for a superpixel segmentation method.
References


