Oriented Light-Field Windows for Scene Flow

Pratul P. Srinivasan[1], Michael W. Tao[1], Ren Ng[1], Ravi Ramamoorthi[2]
{pratul,mtao,ren}@eecs.berkeley.edu, ravir@cs.ucsd.edu
[1]University of California, Berkeley, [2]University of California, San Diego

Abstract

2D spatial image windows are used for comparing pixel values in computer vision applications such as correspondence for optical flow and 3D reconstruction, bilateral filtering, and image segmentation. However, pixel window comparisons can suffer from varying defocus blur and perspective at different depths, and can also lead to a loss of precision. In this paper, we leverage the recent use of light-field cameras to propose alternative oriented light-field windows *that enable more robust and accurate pixel comparisons. For Lambertian surfaces focused to the correct depth, the 2D distribution of angular rays from a pixel remains consistent. We build on this idea to develop an oriented 4D light-field window that accounts for shearing (depth), translation (matching), and windowing. Our main application is to scene flow, a generalization of optical flow to the 3D vector field describing the motion of each point in the scene. We show significant benefits of oriented light-field windows over standard 2D spatial windows. We also demonstrate additional applications of oriented light-field windows for bilateral filtering and image segmentation.*

1. Introduction

Pixel value comparisons are used for a variety of computer vision applications, including finding correspondences between images for 3D reconstruction, enforcing brightness consistency for calculating optical and scene flow, calculating weights for adaptive filters such as bilateral filters, and determining pixel similarity for image segmentation. With conventional images, we can compare pixels by comparing their RGB (or other color space) values. For more robust comparisons, we can also compare 2D pixel windows instead of individual pixels, but this involves a loss of precision. Furthermore, both pixel value comparisons and window comparisons are prone to errors due to the depth of objects in the scene; the windows corresponding to the same scene points at two different depths will appear different due to focal blur and perspective.

We propose a method to represent scene points, that uses the information from 4D light-field images [2, 13], to enable more accurate and robust pixel comparisons (Sec. 3). Each spatial pixel has a 2D distribution of incident angular rays. If focused to the correct effective depth in the scene, all angular rays are consistent for Lambertian surfaces, since all rays converge to the same scene point. When not focused to the correct depth in the scene, the angular rays of a spatial location on the sensor incorporate information from the spatial neighborhood of the scene point. We leverage these insights to develop *oriented 4D light-field windows*, that account for shearing in the light field ray space due to focusing at different depths, translation for pixel matching, and windowing. This is a general representation that in the limit approaches a single spatial pixel with a 2D angular extent, effectively replacing the 2D spatial window, and alleviating issues with loss of precision and defocus blur.

A natural application is the computation of scene flow, since it involves the estimation of corresponding spatial points (Sec. 4). Scene flow [26] is the 3D vector field describing the motion of each scene point over time. It extends the conventional notion of optical flow by also providing the change in depth. Scene flow has many applications in vision, including establishing correspondence, computing camera and object motions, and estimating shape. We propose an algorithm to compute dense non-rigid scene flow from a pair of light-field images, acquired using a consumer light-field camera (Lytro Illum). We demonstrate that oriented light-field windows provide better matching than conventional spatial windows, and derive a 4D light-field matching formula and energy minimization, analogous to traditional 2D brightness consistency.

We further demonstrate oriented light-field windows by using them for filtering images with a bilateral filter and segmenting images (Sec. 5). We show that the results are significantly better than when using traditional 2D images.

2. Previous Work

2.1. Light-Fields in Computer Vision

The 4D light-field is the total spatio-angular distribution of light rays passing through free space, and light-field cam-

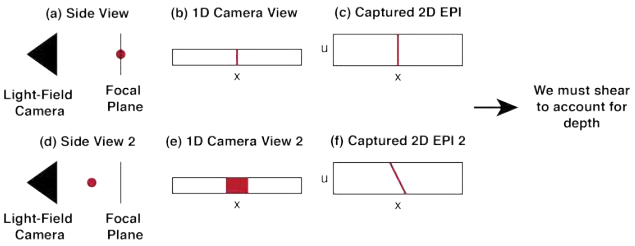

Figure 1. *This setup shows two scenes in a 2D flatland, with side views of (a) and (d) and camera views of (b) and (e). In the first scene, the point is at the focal plane, and in the second scene, it is in front of the focal plane. The conventional camera images in (b) and (e) are different due to perspective and focal blur. We can see in the 2D EPIs (c) and (f) that the change in depth corresponds to a shear in ray-space. To compare these spatial points, we must shear the light-fields to account for the depth.*

eras capture the region of the light-field that exists inside the camera body. Light-field data has enabled many applications such as post-capture image refocusing [13], lens glare artifact reduction [17], and stereo reconstruction from a single capture [2]. Light-field images can be used for passive general depth estimation [21, 23, 25, 28] by taking advantage of the multiple cues, such as defocus and correspondence, that can be obtained from a single shot. In this work, we make use of previous work that uses light-fields for depth estimation and image refocusing, and we introduce a light-field representation for scene points that can be used to significantly improve results in scene flow, bilateral filtering, and image segmentation.

2.2. Scene Flow

Scene flow was introduced by Vedula *et al*. [26], who computed it by first estimating optical flow, the projection of scene flow onto the image plane, in each camera and then using triangulation to calculate a 3D motion field fitted to the estimated optical flows. Later works compute scene flow from stereo [6, 8, 11, 27, 29] or multi-view [5] images at consecutive times, which involves estimating disparity and the change in disparity over time in addition to the optical flow. These works typically use variational approaches to estimate the 3D geometry and flow, either jointly [5, 6, 11, 27] or in a decoupled [29] manner. They employ various methods of regularization and assumptions about the piecewise rigidity of the scene and the flow field. The method of [8] uses local estimation to efficiently compute the 3D geometry and flow. Additionally, other recent methods [9, 10, 12, 16] take advantage of the availability of accurate dense depth information from RGBD cameras to compute scene flow from pairs of RGBD images, typically using the 2D parametrization from RGBD sensors, or, as in [9], a 3D point cloud representation of the scene.

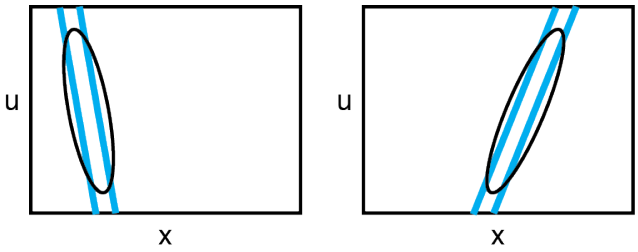

Figure 2. *Oriented light-field windows (black ellipse represents Gaussian weighted window) in a 2D flatland visualization of the 4D EPI, where a scene point (blue) has moved in position and depth between two time steps. We want to match these two windows to detect the same scene point.*

2.3. Bilateral Filtering and Image Segmentation

The bilateral filter [24] has been used widely for many applications such as image denoising, texture removal, and image manipulation [14, 15] to smooth images while respecting edges. The majority of bilateral filtering techniques use the RGB or CIE-Lab color space, and this has been known to lead to bleeding artifacts at edges in certain scenarios. Figure 7 shows that by using oriented light-field windows, desired details and edges are preserved and we improve the robustness of bilateral filters.

Image segmentation [7, 19] is a well-studied problem in computer vision, with the goal of segmenting images into semantically meaningful regions [3], or oversegmenting images into superpixels [1, 18]. Figure 8 shows that we are able to improve the results of the popular SLIC superpixel segmentation algorithm [1] in cases where using traditional pixel values fails. By using oriented light-field windows instead of CIE-Lab color space pixel values, we can better detect texture edges with similar pixel intensity values.

3. Oriented Light-Field Windows

The angular rays of the light-field corresponding to the same point in the scene should be consistent for a Lambertian surface. However, to locate these rays in the light-field, we must account for the shearing effect due to the difference between the depth of the scene point and the light-field focal plane [13]. This is shown in Fig. 1, using a 2D flatland epipolar image (EPI).

We represent each point in the scene as an oriented window in the light-field ray space, as visualized in Fig. 2. The orientation of the window is defined by the shear of the point's effective depth, and the size of the window is defined by spatial and angular Gaussian weights.

Mathematical Definition

As shown in Fig. 3, the oriented light-field window corresponding to a scene point can be computed by a shear operator, a translation operator and a windowing operator, as follows. The shear operator for depth α, as described

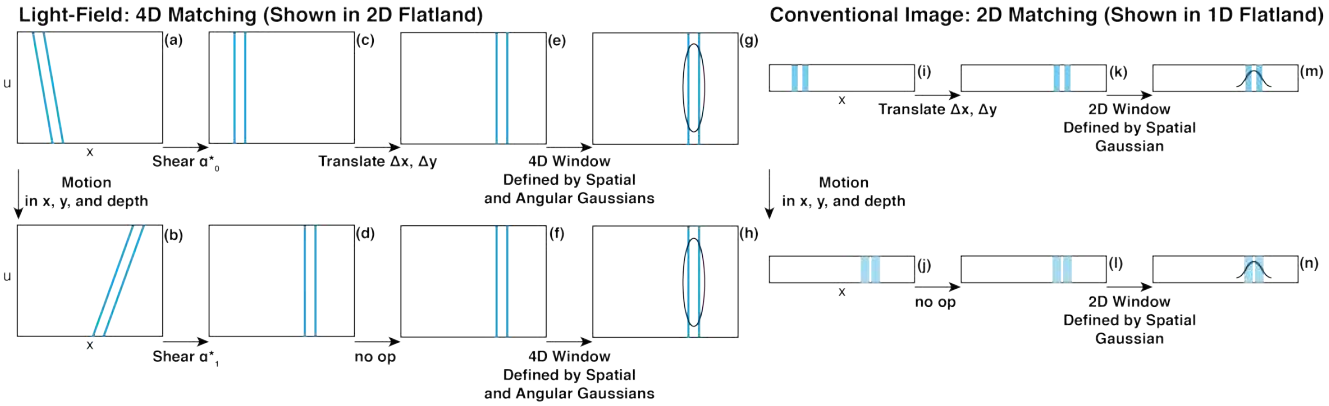

Figure 3. *In this scene, a scene point moves in position and depth. In the 2D flatland visualizations of the 4D EPI (a) and (b), this corresponds to a shear and a shift in position. In the 1D flatland visualizations of the 2D conventional image (i) and (j), this corresponds to a focal blur and a shift in position. To match these two scene points using 4D oriented light-field windows, we shear as in (c) and (d), translate as in (e) and (f), compute the 4D window defined by spatial and angular Gaussian weights as in (g) and (h), and then integrate the difference between the two oriented light-field windows. In contrast, to match these two scene points using 2D image windows, we translate as in (k) and (l), compute the 2D window defined by spatial Gaussian weights as in (m) and (n), and integrate the difference between the two spatial windows.*

in relation to the camera parametrization planes and scene depths in [13], is:

$$\begin{aligned} S_\alpha[L] &\equiv L_\alpha(x,y,u,v) \\ &= L\left(x + u(1-\frac{1}{\alpha}), y + v(1-\frac{1}{\alpha}), u, v\right) \end{aligned} \tag{1}$$

The (spatial) translation operator is defined as usual as:

$$T_{\Delta x,\Delta y}[L] \equiv L_{\Delta x,\Delta y}(x,y,u,v) = L(x+\Delta x, y+\Delta y, u, v) \tag{2}$$

The windowing operator is defined as:

$$W[L] = L(x,y,u,v)\mathcal{N}(x,y;\sigma_{xy}^2)\mathcal{N}(u,v;\sigma_{uv}^2), \tag{3}$$

where $\mathcal{N}(s,t;\sigma^2)$ represents a 2D Gaussian distribution on the st plane, centered at the origin, with variance σ^2 in each dimension, i.e.

$$\mathcal{N}(s,t;\sigma^2) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}(s^2+t^2)\right)$$

Using these operators, the full oriented light-field window operator for a scene point at position (x_0, y_0, α) is:

$$P_{\alpha,x_0,y_0}(x,y,u,v) \equiv (W \circ T_{x_0,y_0} \circ S_\alpha)\,[L], \tag{4}$$

where $P_{\alpha,x_0,y_0}(x,y,u,v)$ can be written explicitly as:

$$\begin{aligned} &P_{\alpha,x_0,y_0}(x,y,u,v) = \mathcal{N}(x,y;\sigma_{xy}^2)\mathcal{N}(u,v;\sigma_{uv}^2)\times \\ &L\left(x + x_0 + u(1-1/\alpha), y + y_0 + v(1-1/\alpha), u, v\right). \end{aligned} \tag{5}$$

Matching

A suitable application of oriented light-field windows is to match pixels between two images, as discussed in the next section on scene flow. In this case, we compare two oriented light-field windows, for example P_{α_0,x_0,y_0} and P_{α_1,x_1,y_1}, computing the sum of squared differences or equivalent metric over the entire 4D window. The process of matching oriented light-field windows versus standard spatial windows is visualized in Fig. 3.

Limiting Cases

The above expression defines the general oriented 4D light-field window. By taking appropriate limits, we can reduce it to 2D spatial or 2D (oriented) angular windows. First, consider $\sigma_{uv} \to 0$, so that we restrict ourselves to $(u,v) = (0,0)$. This reduces to a conventional 2D spatial window in the central pinhole view,

$$P_{x_0,y_0}^{\text{spatial}}(x,y) = L\,(x+x_0, y+y_0, 0, 0)\,\mathcal{N}(x,y;\sigma_{xy}^2). \tag{6}$$

If we take the limit of $\sigma_{uv} \to \infty$ and $\sigma_{xy} \to 0$, we weight all angular directions equally, and the oriented light-field window approaches an in-focus raw light-field image of the point. This is the 2D set of angular directions for a given spatial pixel,

$$\begin{aligned} &P_{\alpha,x_0,y_0}^{\text{angular}}(u,v) = \\ &L\left(x_0 + u(1-\frac{1}{\alpha}), y_0 + v(1-\frac{1}{\alpha}), u, v\right). \end{aligned} \tag{7}$$

In practice, we often use these parameters, which ensures the maximum precision (no spatial extent for windows) and is also efficient (only 2D light-field windows are matched). However, we emphasize that our formulation allows for matching of general 4D oriented light-field windows.

4. Scene Flow

We now develop our algorithm for light-field scene flow. We describe a simple extension to the standard 2D brightness consistency notion, formulate the light-field scene flow calculation as an energy minimization, and develop the optimization and regularization to compute the scene flow.

4.1. Light-Field Brightness Consistency

Scene flow $F(x, y)$ can be described by a 3D vector field,

$$F(x_0, y_0) = (\Delta x, \Delta y, \Delta\alpha) \tag{8}$$

where (x_0, y_0) are pixel coordinates of a 3D point at an effective depth (shear) α_0 in one frame, α_1 in the next frame, and $(\Delta x, \Delta y, \Delta\alpha)$ are the offsets (scene flow) between the frames, with $\alpha_1 = \alpha_0 + \Delta\alpha$.

The traditional brightness consistency assumption in optical flow is that the intensity in both images should be equal for the same point in the scene, and traditionally spatial pixel windows have been matched. By analogy, we assume that the oriented light-field windows are equivalent for the same point in the scene. That is, for all (x, y, u, v)

$$\begin{aligned} &P_{\alpha_0,x_0,y_0}(x, y, u, v, t) = \\ &\qquad P_{\alpha_1,x_0+\Delta x,y_0+\Delta y}(x, y, u, v, t + 1), \end{aligned} \tag{9}$$

where we add a variable t to account for different frames.

4.2. Scene Flow Formulation

We now proceed to solve for scene flow by minimizing an energy function E that includes both a data term E_D and a smoothness term E_S,

$$\begin{aligned} F(x_0, y_0) = \underset{\Delta x,\Delta y,\alpha_0,\alpha_1}{\operatorname{argmin}} \big(&E_D(x_0, y_0, \Delta x, \Delta y, \alpha_0, \alpha_1) \\ &+E_S(x_0, y_0, \Delta x, \Delta y, \alpha_0, \alpha_1)\big). \end{aligned} \tag{10}$$

This formulation also solves for the depths of each scene point in both images, α_0 and α_1. The smoothness term E_S will be discussed in the next sub-section on regularization. In this section, we focus on the data term:

$$\begin{aligned} &E_D(x_0, y_0, \Delta x, \Delta y, \alpha_0, \alpha_1) = \\ &\int \big(P_{\alpha_0,x_0,y_0}(x, y, u, v, t) \\ &- P_{\alpha_1,x_0+\Delta x,y_0+\Delta y}(x, y, u, v, t + 1)\big)^2 \, dx \, dy \, du \, dv. \end{aligned} \tag{11}$$

In practice, the integral will be a discrete summation over both the spatial and the angular domain. Note that this formulation applies to both light-field images and traditional images. As $\sigma_{uv} \to 0$, the angular dimension collapses and the equation reduces to a traditional optical flow formulation with P^{spatial} that just compares 2D pixel windows of pixels from the two central pinhole images.

For robustness to errors in the initial depth estimation, we also include integration over a range of shears, centered at the estimated effective depth of the scene point, when computing our data term.

4.3. Regularization

We use regularization in our scene flow method to enforce a piecewise smooth flow field and propagate accurate flow estimates to areas with low confidence and no local signal. We use total variation regularization, which penalizes the integral of the absolute gradient of the flow field. In the energy minimization framework, our smoothness regularization term is:

$$\begin{aligned} &E_S(x_0, y_0, \Delta x, \Delta y, \alpha_0, \alpha_1) = \\ &\lambda C_F(x_0, y_0) O(x, y)(|\nabla(\Delta x)| + |\nabla(\Delta y)|) + \\ &\gamma C_D(x_0, y_0)(|\nabla(\Delta\alpha)|), \end{aligned} \tag{12}$$

where C_F and C_D are confidence measures for the optical flow and depth estimations, and O is a confidence measure for the occlusion state of a pixel. We discuss how these measures are computed in the next sub-section. In our implementation, we use $\lambda = 0.002$ and $\gamma = 0.1$.

4.4. Scene Flow Estimation

Searching for the scene flow $(\Delta x, \Delta y, \Delta\alpha)$ that minimizes Eq. 11 for every pixel is expensive due to the large search space. We constrain the search space by decoupling the depth and optical flow estimations, using the method from Tao *et al*. [21] to estimate the depths in both light-field images as well as calculate a depth estimation confidence measure $C_D(x_0, y_0)$.

We then compute $(\Delta x, \Delta y)$ by locally searching for the minimum energy within a radius around every pixel, as in the SimpleFlow algorithm [22], for efficiency due to our high-dimensional data. We compute the confidence $C_F(x_0, y_0)$ in $(\Delta x, \Delta y)$ as the minimum subtracted from the mean data energy within the search radius for each pixel.

$$\begin{aligned} C_F(x_0, y_0) = &\underset{(\Delta x,\Delta y)\in N}{\operatorname{mean}} E_D(x_0, y_0, \Delta x, \Delta y, \alpha_0, \alpha_1) - \\ &\min_{(\Delta x,\Delta y)\in N} E_D(x_0, y_0, \Delta x, \Delta y, \alpha_0, \alpha_1) \end{aligned} \tag{13}$$

where N is the set of $(\Delta x, \Delta y)$ within the search radius. This confidence is used in our regularization to increase the smoothness term coefficient for less confident pixels.

We estimate the likelihood that a pixel is not located at an occlusion boundary, $O(x, y)$, by measuring the consistency between the forward and backward optical flows. We use this in our regularization to increase the smoothness term coefficient for pixels that are more likely to be occluded.

The optical flow is optimized over a multiscale pyramid with warping between pyramid levels, resulting in a coarse-

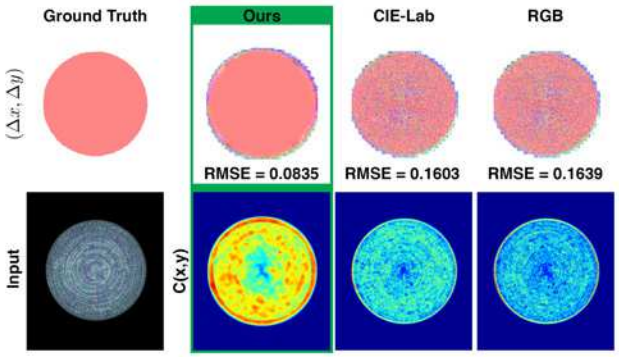

Figure 4. *Local matching optical flow results (one pyramid level, without regularization) calculated for a synthetic sphere translated to the right. Optical flow results are visualized with the Middlebury color code, and the confidence values are visualized with a cold-to-warm color code, where warmer colors signify higher confidence. Using oriented light-field windows provides more accurate and less noisy optical flow results, with higher confidence.*

to-fine strategy that allows the estimation of large displacements. We median filter the intermediate flow results after each warping, as discussed in [20], to remove outliers.

4.5. Results

To validate oriented light-field windows and their use in computing scene flow, we evaluated our algorithm against other state-of-the-art scene flow and optical flow algorithms on both synthetic and real world scenes containing a variety of shapes with motions including rotations and changes in depth. We encourage readers to refer to our supplementary material for more extensive comparisons and examples.

Synthetic Scenes

Figure 4 shows a comparison of local optical flow search results (one pyramid level, without regularization) on a translating synthetic diffuse sphere, using the RGB pixel values from central pinhole images extracted from the light-field, the same pixel values in the CIE-Lab color space, and oriented light-field windows. All RMSE values are normalized by the maximum possible flow error, so the possible RMSE values range from 0 to 1. Using oriented light-field windows yields significantly more accurate and less noisy results with higher confidence, due to the ability of oriented light-field windows to discriminate between scene points with similar pixel intensity values.

Figure 5 shows a synthetic experiment for parameter validation and analysis of the tradeoff between using spatial and angular information in the 4D oriented light-field window. First, note that if we fix σ_{xy}, the RMSE decreases as the angular variance increases from $\sigma_{uv} \rightarrow 0$ (a 2D spatial window) to essentially considering all directions equally. This underlines the benefits of 4D light-field windows over 2D pixel windows. Next, consider spatial variance. For a 2D pixel window ($\sigma_{uv} \rightarrow 0$), the error is quite sensitive to the spatial window size, trading off noise and precision as expected. However, for oriented 4D light-field windows considering all angular directions, the error is relatively insensitive to σ_{xy}. In this example, we found a small spatial window, corresponding to $\sigma_{xy} = 5.45$, to be optimal. In practice, we use a spatial variance of zero and the optimal (largest) angular variance $\sigma_{uv} = 4.09$ for efficiency, since the benefit of using a larger spatial window is negligible.

	Table	Cow	Teddy
Ours	**0.171**	**0.079**	0.409
SRSF	0.353	0.248	0.433
MDP-Flow2	0.210	0.102	0.421
Classic+NL	0.199	0.199	**0.395**
Classic+NL, HR	0.192	0.151	0.416

Table 1. *RMSE values for our algorithm evaluated against state-of-the-art scene flow and optical flow algorithms on three synthetically generated scenes. Example results of all the algorithms on the table scene and our algorithm on the cow and teddy scenes are in Fig. 6. The lowest RMSE for each scene is in bold, and our algorithm is either the top performer (cow and table) or very close in RMSE to the top performer (teddy).*

Figure 6 shows our results for synthetic scenes compared to state-of-the-art scene and optical flow algorithms. We show example results for all algorithms on the table scene, and our algorithm on the cow and teddy scenes. Each scene has one moving Lambertian object with a complex shape, rendered using a single point light source. We compare optical flow results to the Semi-Rigid Scene Flow (SRSF) algorithm of Quiroga *et al*. [16], the Classic+NL optical flow algorithm of Sun *et al*. [20], and the Motion Detail Preserving Optical Flow (MDP-Flow2) algorithm of Xu *et al*. [30], a top performer on the Middlebury optical flow benchmark [4]. We provide both optical flow algorithms and the SRSF algorithm with the central pinhole view from each light-field image as input. As an additional comparison, we provide the Classic+NL method with high-resolution (HR) pinhole images with the spatial resolution equal to our light-field sensor resolution. The MDP-Flow2 method ran out of memory (Intel Core i7 machine with 32 GB RAM) when given the HR images. The SRSF algorithm is designed for RGBD data from a sensor such as the Microsoft Kinect, and requires a dense accurate depth estimation for each input frame. In order to best satisfy this input requirement given our light-field images, we provide the input depth estimation calculated from the light-field images that we use in our algorithm. We do not compare our algorithm to stereoscopic scene flow algorithms such as [6, 11, 27, 29], because they are meant to use pairs of stereo images with wide baselines, and do not perform well when instead given pairs of pinhole images extracted from a light-field camera.

Table 1 shows RMSE values for the algorithms on all three scenes. Since only our algorithm and the SRSF algorithm compute the flow in depth, we only quantitatively

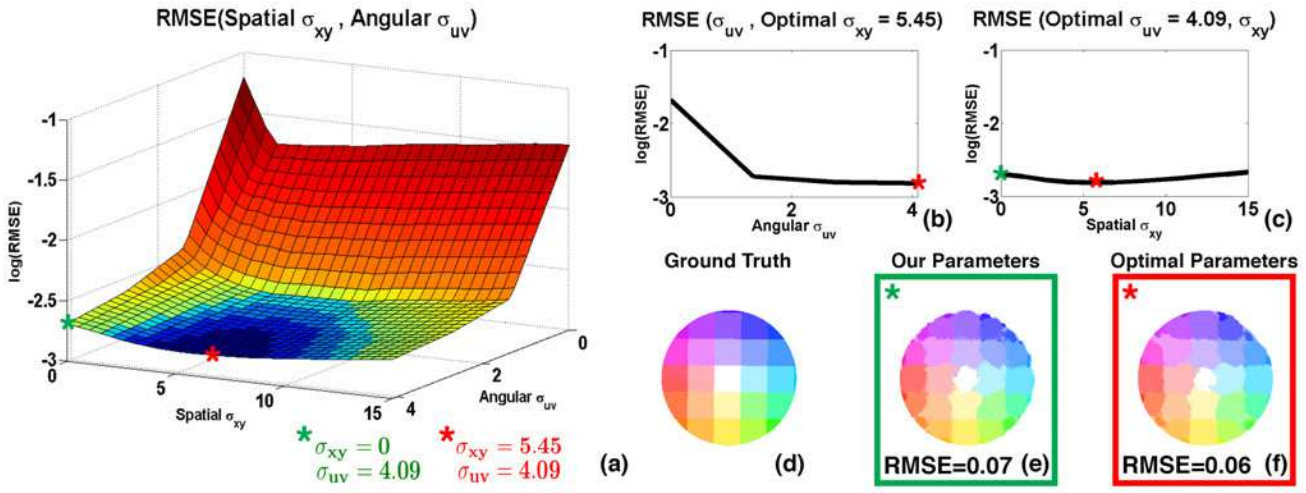

Figure 5. *RMSE for a synthetic sphere rotating 1.2°. In the figure, we show the RMSE as a function of the spatial and angular variances (a), and the slices of the graph at the optimal spatial (b) and angular variances (c) for our synthetic example. The location of the optimal parameters is marked as a red asterisk on (a), and the ground truth local optical flow estimation (d) and results using the optimal (e) and our practical implementation parameters (f) (one pyramid level, without regularization) are shown. The optical flow results look discretized because integer-valued local optical flow estimation has no subpixel flow estimation. With oriented light-field windows, results closely resemble the ground truth. From (a), the RMSE decreases as the angular variance increases up to the maximum provided by the Lytro Illum camera. When $\sigma_{uv} \to 0$, there is clearly an optimal σ_{xy}. However, when we use the optimal σ_{uv}, the benefit of using a larger σ_{xy} is negligible. In practice, we use $\sigma_{xy} = 0$ and the optimal (largest) $\sigma_{uv} = 4.09$ for improved efficiency, and (e) and (f) show that there is a negligible difference between using the optimal and practical implementation parameters.*

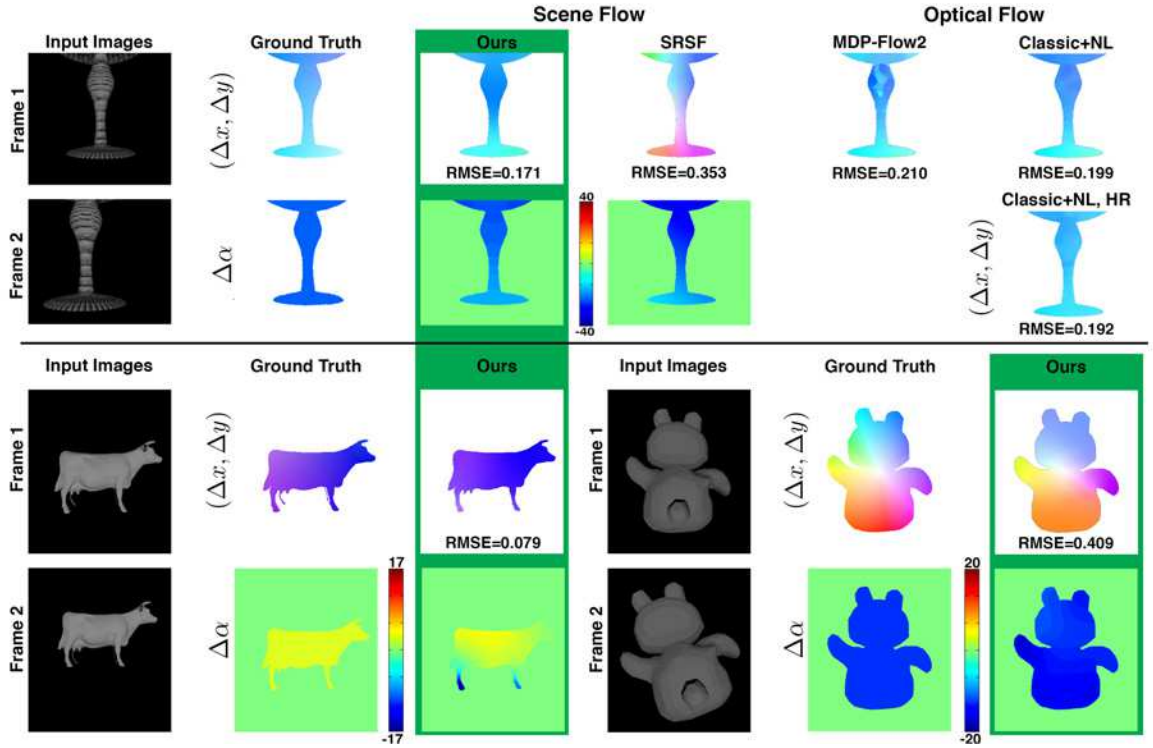

Figure 6. *We compare the results of our algorithm to the SRSF [16] scene flow algorithm and the MDP-Flow2 [30] and Classic+NL [20] optical flow algorithms on three synthetic scenes, and the RMSE values can be seen in Table 1. For each example, the optical flow is visualized with the Middlebury color code, and the flow in depth is visualized with the cold-to-warm color code in the color bars. The three scenes have different motions: the cow moves up and away from the camera, the table moves to the left and towards the camera, and the teddy rotates counterclockwise and moves towards the camera. Our scene flow results are more accurate than others' in most cases.*

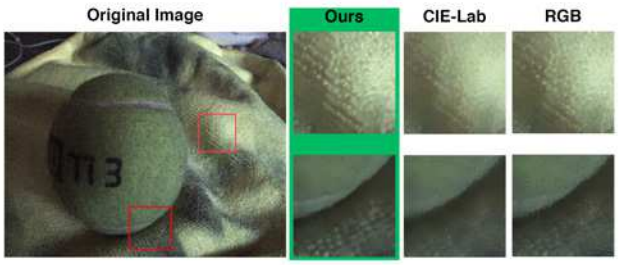

Figure 7. *We compare the results of bilateral filtering an image using RGB or CIE-Lab to those using oriented light-field windows, which are better for preserving textures such as the towel and edges such as that between the felt and rubber.*

compare the RMSE for the optical flow results of the algorithms. As evident from the RMSE results, our algorithm is either the most accurate or very close to the most accurate. Looking at the qualitative results in Fig. 6, we can see that our algorithm is able to accurately estimate the scene flow in scenes where objects rotate and move in depth. Additionally, the Classic+NL HR results show that the decreased spatial resolution of a light-field camera does not significantly impact the flow estimation accuracy.

Real World Scenes

Figure 9 shows results of our scene flow algorithm on natural images, captured with a Lytro Illum camera with 49 sub-aperture images. We compare our scene flow results to the SRSF algorithm, and our optical flow results to the Classic+NL and MDP-Flow2 algorithms. Oriented light-field windows enable significantly more accurate scene flow computation. We correctly estimate the scene flow for significant depth changes, such as in the scenes with the hands and the penguin toy, as well for scenes with similarly colored objects, such as the scene with the tennis ball on a towel. Note that we better capture the motion contours of the hands, tennis ball, and penguin toy, and avoid the patchy appearance of the scene flow from other algorithms.

5. Other Applications

5.1. Bilateral Filtering

Bilateral filter weights are determined by both the spatial closeness and the photometric closeness. Instead of photometric closeness, which is typically measured as Euclidean distance in the RGB or CIE-Lab color space, we use the Euclidean distance between oriented light-field windows at a range of shears. Figure 7 shows that while all methods are able to denoise the image, using oriented light-field windows preserves textures and edges that are similar in pixel value due to similar colors, low illumination, and noise.

5.2. Image Segmentation

Superpixel image segmentation is used as a building block in algorithms such as semantic segmentation and motion estimation. We adapt the SLIC superpixel segmentation algorithm [1] to use oriented light-field windows at a range of shears instead of CIE-Lab pixel values. Figure 8 shows that using oriented light-field windows enables better superpixel segmentation at edges that are similar in pixel value due to similar colors, low illumination, and noise.

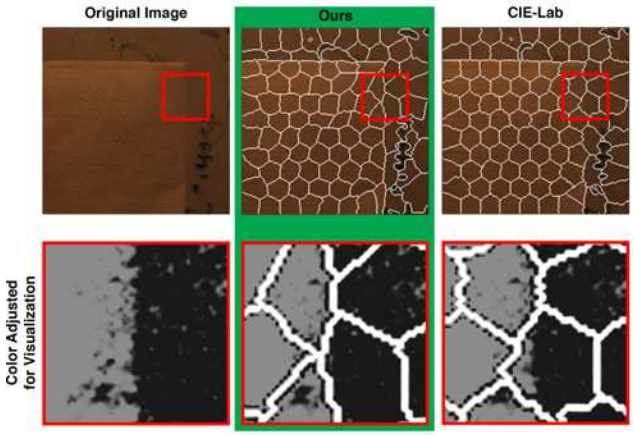

Figure 8. *We compare the results of the SLIC superpixel segmentation algorithm [1] using the default CIE-Lab pixel values and oriented light-field windows. We adjust the color for the figure insets to highlight the napkin edge for better visualization. Oriented light-field windows are better for detecting the edge between the paper napkin and the background wall.*

6. Conclusion and Future Work

We proposed *oriented light-field windows*, a novel accurate and robust method of pixel comparison using light-field images. Oriented 4D light-field windows represent scene points by accounting for shearing (depth), translation (matching), and windowing. We apply oriented light-field windows to compute scene flow, and show significant benefits over standard 2D spatial windows by analyzing the tradeoff between the spatial and angular variance of the windows. We further demonstrate the benefits of using oriented light-field windows by evaluating our algorithm against state-of-the-art methods in scene and optical flow. Finally, we demonstrate applications in bilateral filtering and image segmentation, where we show that we are able to better detect and preserve edges.

Oriented light-field windows are a general way to describe scene points, and can be used to formulate many other problems, such as finding correspondences for reconstruction and edit propagation in video, for light-fields. This work builds the foundation for the use of light-field images for many computer vision and graphics applications.

Acknowledgments

This work was supported in part by ONR grant N00014-15-1-2013, funding from Nokia and Intel, and support by Sony to the UC San Diego Center for Visual Computing. Some of this work was done while Ren Ng was at Lytro, Inc.

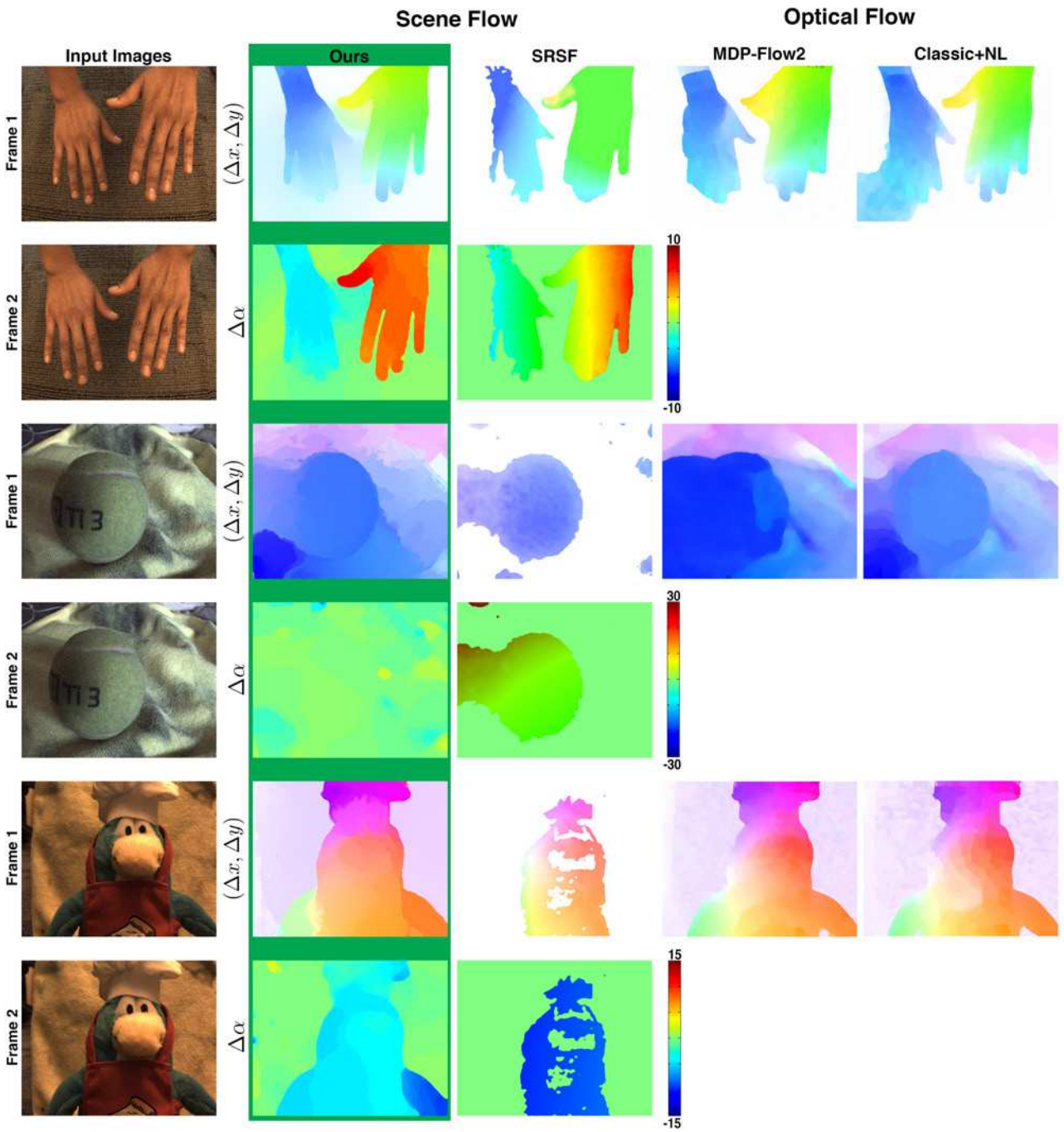

Figure 9. *We compare our scene flow results against the SRSF [16] RGBD scene flow algorithm, and the MDP-Flow2 [30] and Classic+NL [20] optical flow algorithms. For each example, the first row contains the optical flow $(\Delta x, \Delta y)$ visualized with the Middlebury color code, and the second row contains the flow in depth $\Delta\alpha$ visualized with the cold-to-warm color code in the color bars, where the green color corresponds to zero depth motion, cooler colors correspond to depth motion toward the camera, and warmer colors correspond to depth motion away from the camera. In the top two rows, we have an example where two hands move in depth. We can see that our algorithm is able to accurately estimate the scene flow, even at occlusion boundaries, with accurate contours around the hand borders. In the second and third rows, we have an example where a tennis ball is on a towel, and they both shift slightly without significant depth motion. We can see that our algorithm is able to accurately estimate the scene flow, adheres to the border of the ball, and correctly calculates no significant changes in depth. In the bottom row, we have an example of a toy penguin that moves towards the camera. We can see that our algorithm correctly estimates the scene flow and adheres to the object borders.*

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012. 2, 7

[2] E. Adelson and J. Wang. Single lens stereo with a plenoptic camera. *PAMI*, 1992. 1, 2

[3] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *CVPR*, 2012. 2

[4] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 5

[5] T. Basha, Y. Moses, and N. Kiryati. Multi-view scene flow estimation: a view centered variational approach. *IJCV*, 2012. 2

[6] J. Cech, J. Sanchez-Riera, and R. Horaud. Scene flow estimation by growing correspondence seeds. In *CVPR*, 2011. 2, 5

[7] P. Felzenswalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 2

[8] M. Gong. Real-time joint disparity and disparity flow estimation on programmable graphics hardware. *CVIU*, 2008. 2

[9] S. Hadfield and R. Bowden. Scene particles: unregularized particle based scene flow estimation. *PAMI*, 2014. 2

[10] E. Herbst, X. Ren, and D. Fox. RGB-D flow: dense 3-D motion estimation using color and depth. In *ICRA*, 2013. 2

[11] F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *ICCV*, 2007. 2, 5

[12] A. Letouzey, B. Petit, and E. Boyer. Surface flow from depth and color images. In *BMVC*, 2011. 2

[13] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photographhy with a hand-held plenoptic camera. *CSTR 2005-02*, 2005. 1, 2, 3

[14] S. Paris, S. Hasinoff, and J. Kautz. Local laplacian filters: edge-aware image processing with a laplacian pyramid. *ACM Transactions on Graphics*, 2011. 2

[15] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand. Bilateral filtering: theory and applications. In *Foundations and Trends in Computer Graphics and Vision*, 2008. 2

[16] J. Quiroga, T. Brox, F. Devernay, and J. Crowley. Dense semi-rigid scene flow estimation from RGBD images. In *ECCV*, 2014. 2, 5, 6, 8

[17] R. Raskar, A. Agrawal, C. Wilson, and A. Veeraraghavan. Glare aware photography: 4D ray sampling for reducing glare effects of camera lenses. In *ACM SIGGRAPH*, 2008. 2

[18] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, 2003. 2

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 1997. 2

[20] D. Sun, S. Roth, and M. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010. 5, 6, 8

[21] M. Tao, S. Hadap, J. Malik, and R. Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *ICCV*, 2013. 2, 4

[22] M. Tao, B. J., P. Kohli, and S. Paris. SimpleFlow: a non-iterative, sub linear optical flow algorithm. In *Eurographics*, 2012. 4

[23] M. Tao, T.-C. Wang, J. Malik, and R. Ramamoorthi. Depth estimation for glossy surfaces with light-field cameras. In *ECCV LF4CV*, 2014. 2

[24] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998. 2

[25] I. Tosic and K. Berkner. Light field scale-depth space transform for dense depth estimation. In *CVPR Workshop on Computational Cameras and Displays*, 2014. 2

[26] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *ICCV*, 1999. 1, 2

[27] C. Vogel, K. Schindler, and S. Roth. Piecewise rigid scene flow. In *ICCV*, 2013. 2, 5

[28] S. Wanner and B. Goldluecke. Globally consistent depth labeling of 4D light fields. In *CVPR*, 2012. 2

[29] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers. Stereoscopic scene flow computation for 3D motion understanding. *IJCV*, 2010. 2, 5

[30] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *PAMI*, 2012. 5, 6, 8