Depth from Defocus in the Wild

Huixuan Tang[1] Scott Cohen[2] Brian Price[2] Stephen Schiller[2] Kiriakos N. Kutulakos[1]
[1] University of Toronto [2] Adobe Research

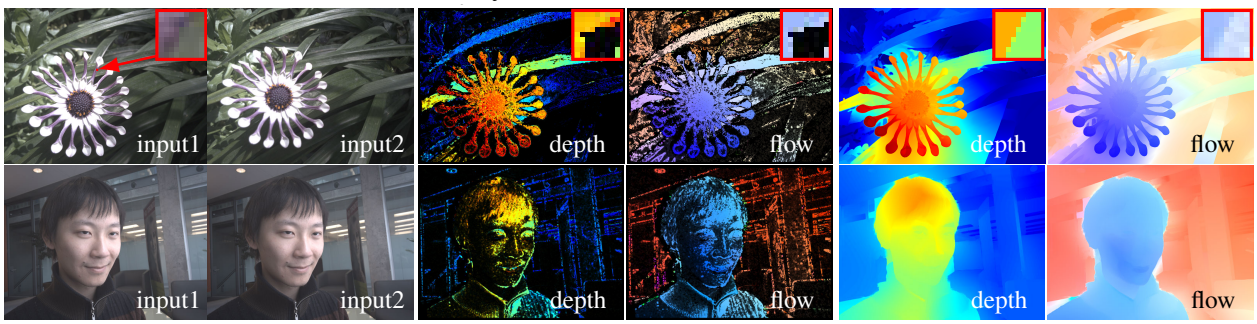


Figure 1: From just two Nexus N5 cellphone photos exhibiting tiny defocus blur and significant inter-frame motion we jointly estimate likelihood functions for local depth and 2D flow. These likelihoods are computed independently from very small, 9×9-pixel patches (see inset and zoom far into the electronic copy for actual size). Of these, only a sparse subset is associated with high-confidence depths and flows (shown color-coded in the middle). We use these very sparse local estimates to infer dense depth and flow in a way that yields sharp boundaries and respects depth-order relations (rightmost images). Note the sharp boundaries and thin structures preserved in the flower's dense depth map; the spatially-varying leaf deformations captured in its flow map; the depth and flow recovered from a pair of selfies with little texture; and the flow around the subject's jaw, caused by a slight change in facial expression.

Abstract

We consider the problem of two-frame depth from defocus in conditions unsuitable for existing methods yet typical of everyday photography: a non-stationary scene, a handheld cellphone camera, a small aperture, and sparse scene texture. The key idea of our approach is to combine local estimation of depth and flow in very small patches with a global analysis of image content—3D surfaces, deformations, figure-ground relations, textures. To enable local estimation we (1) derive novel defocus-equalization filters that induce brightness constancy across frames and (2) impose a tight upper bound on defocus blur—just three pixels in radius—by appropriately refocusing the camera for the second input frame. For global analysis we use a novel spline-based scene representation that can propagate depth and flow across large irregularly-shaped regions. Our experiments show that this combination preserves sharp boundaries and yields good depth and flow maps in the face of significant noise, non-rigidity, and data sparsity.

1. Introduction

The technique of *depth from defocus*—recovering a depth map from two differently-focused images of a scene—has been studied extensively in computer vision for almost three decades [1, 9, 19, 22, 31, 36]. Although the basic theory behind this technique is well known, depth from defocus (DFD) has found limited use in practice because it is broadly understood to require static scenes, dense surface texture, and images with significant defocus blur. These assumptions rarely hold "in the wild," where cameras are handheld and often on a cellphone; lens apertures are small; surfaces in the scene may move or deform; and scene texture is generally unconstrained.

In this paper we show how to compute DFD under such challenging conditions from minimal input: two cellphone photos of an unrestricted scene, having visually-imperceptible defocus blur and captured in rapid succession (Figure 1). This approach stands in sharp contrast to recent passive depth estimation techniques for mobile cameras (*e.g.*, depth from focal stacks [24, 26] and structure from motion [8, 35]) which require dozens of photos and prolonged movement to capture a reliable depth map, and cannot handle non-rigid scene deformation.

More specifically, we tackle the following challenges:
- **tiny blur:** cellphone cameras have small apertures that produce very little defocus blur relative to the image size;
- **scene deformation:** since motion between a pair of shots is often unavoidable, 2D flow estimation and DFD are tightly coupled and cannot be solved independently;
- **sparse defocus:** defocus blur is only observable in the neighborhood of strong textures and brightness edges, both of which may be sparse in a general scene;
- **depth discontinuities:** thin structures and depth discontinuities occur often and must be handled robustly;
- **figure-ground ambiguities:** even when defocus can be measured at an isolated brightness edge, it may not fully constrain local scene geometry (*e.g.*, surface markings and depth discontinuities are indistinguishable).

scene	camera	ISO	depth range	motion type	flow magnitude
keyboard	Samsung	N/A	1cm-2m	rigid	< 5 pixels
balls	Samsung	N/A	1cm-2m	rigid	< 5 pixels
fruit	Samsung	N/A	5cm-3m	rigid	< 5 pixels
spike	Nexus5	100	8cm-30cm	non-rigid	< 80 pixels
face	Nexus5	180	20cm-∞	non-rigid	< 70 pixels
bagels	Nexus5	222	10cm-30cm	rigid	< 80pixels
flower	Nexus5	100	10cm-30cm	piecewise rigid	< 80 pixels
bell	Nexus5	143	15cm-80cm	piecewise rigid	< 60pixels
potrait	Nexus5	180	20cm-∞	non-rigid	< 150 pixels
patio	Canon7D	100	6m-∞	piecewise rigid	< 50 pixels
stairs	Canon7D	100	2m-∞	non-rigid	< 70 pixels

Table 1: Scenes, cameras and imaging conditions (Figures 1,9,10).

We rely on three novel contributions to achieve this goal. First, and foremost, we observe that rather than being a hindrance, tiny blur can be turned to an advantage because it enables extremely local processing, preserves high spatial frequencies and offers near-optimal depth discrimination. We exploit this observation by controlling camera focus to enforce the *tiny blur condition*: the radius of the defocus blur kernel is at most three pixels for the scene points we wish to reconstruct. This upper bound on defocus covers a wide range of near-field and far-field imaging conditions (Table 1) and is only slightly higher than the optimal blur kernels implied by Schechner and Kiriyati's early work [23]. This makes it possible to perform local DFD and flow estimation by analyzing tiny patches: just 9×9 pixels in size in our megapixel-sized input images.

Second, we derive a novel quadratic likelihood function over depth and 2D flow that correctly accounts for differences in defocus and illumination for each input patch. Computationally, the key step involves applying *depth-specific defocus equalization filters* to the input images to ensure that (1) the two filtered images have exactly the same blur if a depth hypothesis is correct and (2) the likelihood function is not biased toward depths with large blur kernels. This leads to a "local DFD" procedure that can be viewed as a form of patch-based translational flow estimation [4] with a hidden depth variable. Our local DFD method is in contrast to prior work that requires large textured regions to estimate 3D flow reliably [1], assumes the motion is purely due to parallax [16, 29] or estimates coarse 2D flow from focal stacks without accounting for defocus [24, 26]. Our method needs far fewer images; does not require inferring a sharp image; does not impose a global motion model; and allows us to keep track of spatially-varying uncertainty in depth and flow (*i.e.*, due to variations in frequency content from patch to patch). For static scenes and static illumination our local DFD likelihood function reduces to that of Zhou and Nayar's [36], with a flat prior over frequencies.

Third, we conduct a global analysis of image content—textures, 3D surfaces, surface deformations, surface boundaries, figure-ground relations—to turn locally-computed likelihoods into dense flow and depth maps. This approach has been used before for depth cues other than defocus [5–7, 15, 30, 33, 34]. However, the noise, sparsity and ambiguities inherent in DFD require fusion over far greater image distances and irregularly-shaped regions, and require explicit reasoning about figure-ground relations. We introduce a *spline-based piecewise-continuous prior* for this purpose and use discrete-continuous energy minimization to compute maximum-likelihood dense depth and flow. This formulation can be viewed as an extension of semi-dense methods [30, 33, 34] that handles far sparser depth/flow data, accounts for spatially-varying data uncertainty, and does not rely on square-shaped regions for fusion [7]. Moreover, unlike variational DFD methods [10, 18, 21] it enables reasoning about which pixels "share the same surface" [9].

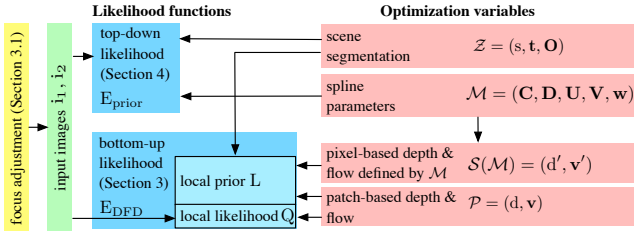


Figure 2: Basic components of our method. Arrows indicate the quantities contributing to each likelihood function. See Figure 7 for the definition of the scene segmentation and spline parameters.

2. Overview

Our method uses two input images taken under controlled focus. The focus setting of the first image is chosen freely; the focus of the second is chosen to satisfy the tiny blur condition (Section 3.1). This functionality is supported by many cellphone and single-reflex (SLR) cameras.[1]

We infer the depth map of the scene and the optical flow field between the two images by optimizing over a set of variables organized in four layers of abstraction (Figure 2). At the lowest level, we represent the depth and flow at every 9×9 patch in the first input image. This defines a patch-based depth map d and flow field $\mathbf{v}$. The depth and flow of each patch is connected to the input images via a local likelihood term Q discussed in Section 3. At the next level of abstraction we represent the depth and flow of pixels individually. This defines a second, pixel-based depth map d' and flow field $\mathbf{v}'$ that are linked probabilistically to the patch-based depth and flow via a local prior term L. Upon convergence, d' and $\mathbf{v}'$ are the output of our method.

The pixel-based depth and flow are obtained by evaluating a piecewise continuous scene model $\mathcal{M}$ on the discrete pixel grid. This model is discussed in Section 4. Each continuous segment in $\mathcal{M}$ corresponds to a smooth surface in the scene, and is expressed as two splines: one for describing depth as a mixture of planes with bounded spatial support, and one for describing flow as a mixture of affine flow fields. The top layer $\mathcal{Z}$ represents the global scene segmentation and consists of (1) a pixel ownership map s that maps pixels to the segment they belong to; (2) a plane ownership map $\mathbf{t}$ that maps planes in the spline model to the segment they

[1] We use the Canon EDSDK (kEdsCameraCommand_DriveLensEvf) and the Android camera2 API (CaptureRequest.LENS_FOCUS_DISTANCE).

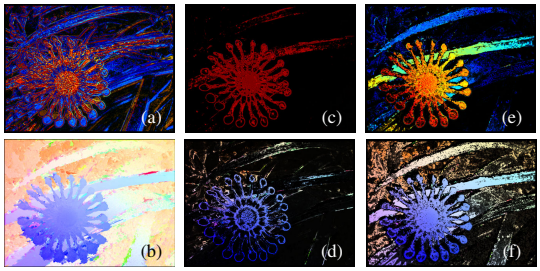


Figure 3: Comparison to alternative local methods (please zoom in). Input images are as in Figure 1 (top row). (a) Depth from the DFD method of [36]. (b) Flow from the method of [3]. (c),(d) Local depth and flow obtained with the same optimization and input parameters as Figure 1 except that filters g_1, g_2 are replaced with those of [25]. (e),(f) Local DFD produces far better results.

belong to; and (3) a matrix $\mathbf{O}$ that describes the occlusion relationship between pairs of segments.

The four levels are coupled probabilistically via two likelihood functions, $\mathrm{E}_{\mathrm{DFD}}$ and $\mathrm{E}_{\mathrm{prior}}$, that capture bottom-up and top-down constraints, respectively. This leads to the following general optimization problem:

$$\min_{\mathcal{P},\mathcal{M},\mathcal{Z}} \left[\mathrm{E}_{\mathrm{DFD}}(\mathcal{P}, \mathcal{S}(\mathcal{M}), \mathcal{Z}) + \mathrm{E}_{\mathrm{prior}}(\mathcal{M}, \mathcal{Z}) \right] . \quad (1)$$

We discuss these likelihood functions in the next two sections. Full details on optimizing Eq. (1) are in [27]. Our code and data can be found in [28].

3. Depth and Flow by Local DFD

No existing method can solve DFD when the scene deforms. In such cases it is not possible to estimate 2D flow without analyzing defocus (brightness constancy violated) and it is not possible to analyze defocus without estimating flow (pixelwise correspondence violated). See Figure 3(a) and (b).

We begin with the problem of estimating the joint likelihood of a depth and flow hypothesis $(d, \boldsymbol{v})$ in a small patch $\Omega(\mathbf{p})$ centered at pixel $\mathbf{p}$ of the first input image. In the following we use homogeneous 2D coordinates to represent $\mathbf{p}$ and express its depth in units of diopters, *i.e.*, units of reciprocal distance from the lens aperture.

Thin-lens model As with most work on DFD, we assume defocus blur is governed by the thin-lens model (Figure 4(a)). This model has four main parameters: the focal length F of the lens; the radius of its circular aperture; and the focus settings f_1 and f_2 of the two input images, expressed as distances from the aperture to each sensor plane. Under this model, an isolated scene point at depth d will appear as a disk in each input image whose radius is proportional to both d and the image's focus setting. This disk defines the point's *blur kernel* in each image, which we denote by k_1^d and k_2^d, respectively. We assume that the thin-lens parameters are known from camera calibration and that the kernels $\mathrm{k}_1^d, \mathrm{k}_2^d$ can be computed for any depth d. See [27] for their analytical expressions and calibration procedure.

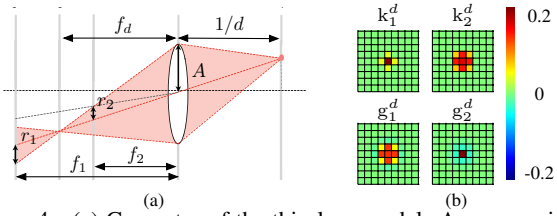


Figure 4: (a) Geometry of the thin-lens model. A scene point at depth d is in focus at distance f_d behind the lens. Its blur kernel in the input images has radius r_1 and r_2, respectively. (b) Defocus kernels and their associated defocus-equalization filters. These filters contain negative values and thus do not merely blur the input. The blur kernels shown have radius 0.6 and 1.4 pixels, respectively, and are inside the 9×9-pixel patch we use for local DFD.

Image formation We consider the input images i_1 and i_2 to be proportional to sensor irradiance and to be corrupted by additive i.i.d. Gaussian noise of known variance σ_i^2.

Unbiased defocus-equalization filters A major barrier to motion estimation across a pair of differently-focused images is that the appearance of scene points will differ across them. This violates the brightness constancy assumption [1]. To overcome it, we derive two novel filters g_1^d and g_2^d that yield a well-founded likelihood function for depth and flow. In particular, we prove the following:

***Proposition* 1** *If* i_1 *and* i_2 *are fronto-parallel image patches related by a 2D translation* $\boldsymbol{v}$ *and an intensity scaling* α *that is sufficiently close to one, the image error*

$$(\mathrm{i}_1 * \mathrm{g}_1^d)(\mathbf{p}) - \alpha \cdot (\mathrm{i}_2 * \mathrm{g}_2^d)(\mathbf{p} + \boldsymbol{v}) \quad (2)$$

follows the same distribution as the noise in i_1 *and* i_2*. The defocus-equalization filters* g_1^d *and* g_2^d *are defined as*

$$\begin{aligned} \mathrm{g}_1^d &= \mathcal{F}^{-1}\left[\mathcal{F}[\mathrm{k}_2^d] \,/\, \sqrt{\mathcal{F}[\mathrm{k}_1^d]^2 + \mathcal{F}[\mathrm{k}_2^d]^2} \right] \\ \mathrm{g}_2^d &= \mathcal{F}^{-1}\left[\mathcal{F}[\mathrm{k}_1^d] \,/\, \sqrt{\mathcal{F}[\mathrm{k}_1^d]^2 + \mathcal{F}[\mathrm{k}_2^d]^2} \right] \end{aligned} \quad (3)$$

where $\mathcal{F}[], \mathcal{F}^{-1}[]$ *denote the Fourier transform and its inverse.*

See [27] for a proof. Intuitively, the numerators $\mathcal{F}[\mathrm{k}_2^d]$ and $\mathcal{F}[\mathrm{k}_1^d]$ in Eq. (3) ensure brightness constancy by convolving each image with the blur kernel of the other. This is similar to previous blur equalization techniques [16, 20, 25, 32]. The novelty here is in the denominators in Eq. (3). These guarantee that the difference of two defocus-equalized images has the same variance as the distribution of image noise. This avoids frequency-based biases typical of existing methods (see Figures 3(c) and (d)).

Local likelihood term Q Proposition 1 leads directly to a likelihood function for depth d and flow $\boldsymbol{v}$ that is just a sum of squared differences:

$$\begin{aligned} &-\log \Pr(\mathrm{i}_1, \mathrm{i}_2 \mid \mathrm{d}_{\mathbf{p}} = d, \mathbf{v}_{\mathbf{p}} = \boldsymbol{v}) = \\ &\min_{\alpha} \sum_{\mathbf{q} \in \Omega(\mathbf{p})} \frac{[(\mathrm{i}_1 * \mathrm{g}_1^d)(\mathbf{q}) - \alpha \cdot (\mathrm{i}_2 * \mathrm{g}_2^d)(\mathbf{q} + \boldsymbol{v})]^2}{2\sigma_i^2} , \end{aligned} \quad (4)$$

where $\mathrm{d}_{\mathbf{p}}$ and $\mathbf{v}_{\mathbf{p}}$ denote the value of the depth map d and flow field $\mathbf{v}$ at pixel $\mathbf{p}$. The unknown scalar α accounts for illumination change. For any hypothesis $(d, \boldsymbol{v})$, we compute α analytically since the sum in Eq. (4) is quadratic in α.

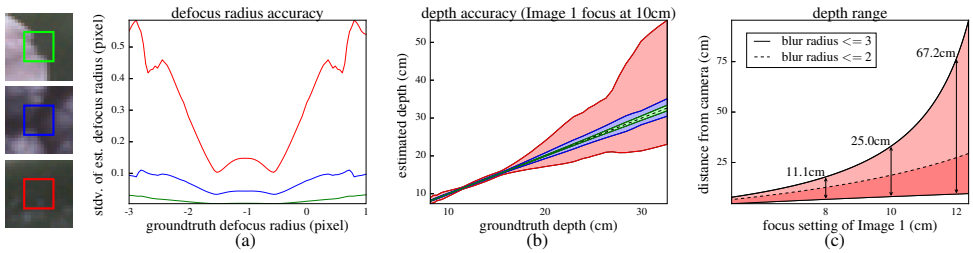


Figure 5: (a) Predicting depth uncertainty for three 9×9-pixel patches taken from the flower photo in Figure 1 (outlined in color on the left). For each patch, we calculate the standard deviation of the defocus kernel's maximum likelihood (ML) estimate as a function of the ground-truth kernel. This amounts to computing the second derivative of Eq. (4) at the ML depth. The plots confirm our intuition that defocus estimation should be much more precise near an edge (green patch) than on patches with weak (blue) or no (red) texture. (b) Taking the Nexus N5's lens parameters into account, it is possible to convert the plot in (a) into a prediction of actual distance errors to expect from DFD on those patches. (c) Enforcing the tiny blur condition. We first focus at the desired distance (point on the x axis) and then set the second focus to maximize the condition's working range (see [27] for the analytic expression). The plots show the working ranges of Schechner and Kiryati's optimality condition (red) and of ours (red and pink).

Although evaluating the likelihood of any given hypothesis $(d, \boldsymbol{v})$ is straightforward, the likelihood in Eq. (4) is not an analytical function of d and $\boldsymbol{v}$. This makes global optimization of Eq. (1) hard. To enable efficient inference, we evaluate the likelihood function at 32 depth samples and 49 flow samples around the maximum-likelihood estimates $\mathrm{d}^*_{\mathbf{p}}$ and $\mathbf{v}^*_{\mathbf{p}}$ of each patch, and fit the following quadratic function:

$$\mathrm{Q}_{\mathbf{p}}(d, \boldsymbol{v}) = \sigma_{\mathbf{p}}^{-2}(d - \mathrm{d}^*_{\mathbf{p}})^2 + (\boldsymbol{v} - \mathbf{v}^*_{\mathbf{p}})^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{p}}^{-1} (\boldsymbol{v} - \mathbf{v}^*_{\mathbf{p}}) + \mathrm{q}_{\mathbf{p}} \quad (5)$$

See [27] for details of the fitting algorithm. The advantage of this approximation is that it gives an estimate of the depth and flow uncertainty of each individual patch in the image: for each pixel $\mathbf{p}$, we obtain the depth variance $\sigma^2_{\mathbf{p}}$ for patch $\Omega(\mathbf{p})$, the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{p}}$ of its flow, and the likelihood $\mathrm{q}_{\mathbf{p}}$. An example is shown in Figures 5(a) and (b).

Local prior term L Each pixel $\mathbf{p}$ in the image belongs to many overlapping patches, not just the one centered at $\mathbf{p}$. These patches may provide conflicting or mutually-reinforcing estimates of depth and flow, may have different variances and covariances (Eq. (5)), and some may even span a depth discontinuity. These interactions must be taken into account when assigning depths and flows to individual pixels. We do this with a local smoothness prior between pixels $\mathbf{q}$ inside a patch $\Omega(\mathbf{p})$:

$$\mathrm{L}_{\mathbf{qp}}(d, \boldsymbol{v}, d', \boldsymbol{v}', \mathcal{Z}) = \begin{cases} \frac{(d-d')^2}{2\sigma_d^2} + \frac{|\boldsymbol{v}-\boldsymbol{v}'|^2}{2\sigma_{\mathbf{v}}^2} & \text{if } \mathbf{q} \in \Omega_f(\mathbf{p}) \\ \tau_o & \text{otherwise} \end{cases} \quad (6)$$

where $d, \boldsymbol{v}$ are the depth and flow of patch $\Omega(\mathbf{p})$; $d', \boldsymbol{v}'$ are those of pixel $\mathbf{q}$; $\Omega_f(\mathbf{p})$ is the set of pixels in $\Omega(\mathbf{p})$ that lie on the front-most surface according to the segmentation $\mathcal{Z}$; and τ_o is a constant that penalizes significant occlusions.

The bottom-up likelihood $\mathrm{E}_{\mathbf{DFD}}$ Together, the local likelihood term (Eq. (5)) and local prior term (Eq. (6)) yield the likelihood of the pixel-based depth and flow map:

$$\mathrm{E}_{\mathrm{DFD}}(\mathcal{P}, \mathcal{S}, \mathcal{Z}) = \sum_{\mathbf{p}} \underbrace{\min(\mathrm{Q}_{\mathbf{p}}(\mathrm{d}_{\mathbf{p}}, \mathbf{v}_{\mathbf{p}}), \tau_i)}_{\text{local likelihood term}} + \sum_{\mathbf{p}} \sum_{\mathbf{q} \in \Omega(\mathbf{p})} \underbrace{\mathrm{L}_{\mathbf{qp}}(\mathrm{d}_{\mathbf{p}}, \mathbf{v}_{\mathbf{p}}, \mathrm{d}'_{\mathbf{p}}, \mathbf{v}'_{\mathbf{p}}, \mathcal{Z})}_{\text{local prior term}}, \quad (7)$$

where the outer summations are over all image pixels. Here τ_i is a constant that reduces the influence of "outlier patches," *i.e.*, patches that contain pixels with very different depth or flow estimates.

3.1. The Tiny Blur Condition

Our bottom-up likelihood function makes no assumptions about the size of the blur kernel or the size of image patches. DFD accuracy, however, does depend on them being very small. To enforce this condition, we actively control the camera's focus setting (Figure 5(c)).

This is justified on both theoretical and empirical grounds. On the theory side, Schechner and Kiryati [23] have shown that optimal depth discrimination for two-frame DFD is achieved for defocus kernels $\mathrm{k}^d_1, \mathrm{k}^d_2$ that (1) differ by exactly one pixel and (2) have an absolute size of less than two pixels. Therefore, refocusing to meet this condition produces more accurate depths. On the practical side, small blurs permit small patches for local DFD. This reduces the co-occurrence of patches and depth discontinuities and allows simple local priors, like that of Eq. (6), to handle them.

While elegant mathematically, Schechner and Kiryati's condition is too restrictive for everyday imaging. This is because it restricts the camera's working range to a fairly narrow range of depths (Figure 5(c), red region). Thus, rather than enforce it exactly, we relax it by choosing focus settings that permit slightly larger kernels (*up to three pixels in radius*) and induce a larger blur difference between them (*exactly two pixels*). This extends the working range considerably (Figure 5(c), red and pink region) without a big impact on error. Figure 6 shows an example.

DFD accuracy degrades gracefully for points outside the range of the tiny blur assumption. Points far outside it will have a large estimated variance ($\sigma^2_{\mathbf{p}}$ in Eq. (5)) and, as a result, will not affect global DFD computations.

Choice of patch size We choose patch size to be 9×9, *i.e.*, three times the maximum blur radius. This is large enough to ensure validity of defocus-equalization filtering and small enough to make within-patch depth discontinuities rare.

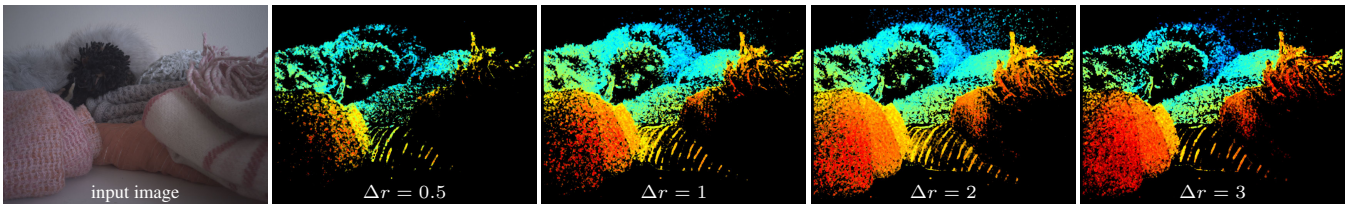


Figure 6: Experimental validation of the tiny blur condition. We capture several DFD image pairs with the Nexus5 camera for a scene in the range $[30\text{cm}, 80\text{cm}]$, one of which is shown on the left. Depth maps on the right show the results of applying Local DFD to pairs of focus settings that induce very similar ($\Delta r \leq 1$) to fairly different ($\Delta r = 3$) blur kernels. The black pixels in the results of Local DFD mark patches with low confidence, *i.e.*, where the depth variance $\sigma^2_{\mathbf{p}}$ is above a threshold. The number of confident patches is highest when the tiny blur condition is met ($\Delta r = 2$) and degrades gracefully when the blur kernels deviate from it.

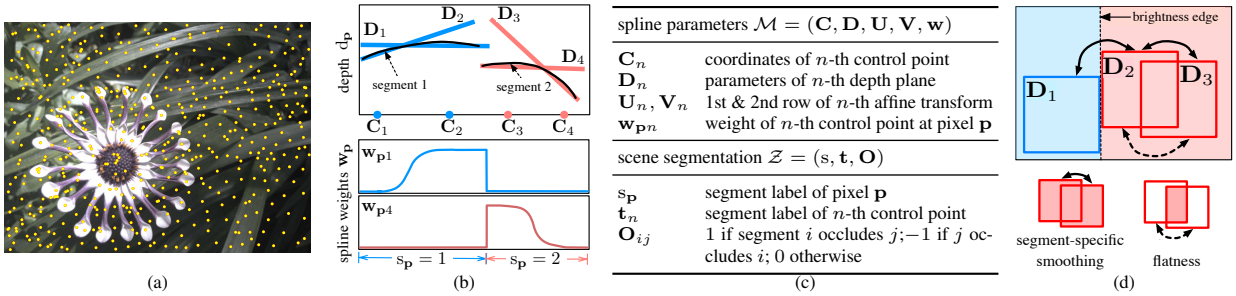


Figure 7: Our spline-based scene representation. (a) Initial control points are shown as yellow dots in the image. (b) Each smooth segment in the scene (black curves) owns a disjoint subset of these control points. Each control point $\mathbf{C}_n$ has an associated depth plane with parameters $\mathbf{D}_n$. The depth at a pixel is expressed as a weighted combination of the depths predicted by these planes. For any given pixel, the weights are non-zero only for control points belonging to the pixel's segment. (c) Spline and segmentation parameters are estimated from the input images. The subscript n denotes the n-th row of a matrix or column vector. (d) Comparison of the flatness prior $\mathrm{E_{flat}}$ and the segment-specific smoothing prior $\mathrm{E_{smo}}$. In this example, the term $\mathrm{E_{flat}}$ does not enforce smoothness between planes $\mathbf{D}_1$ and $\mathbf{D}_2$ because of the brightness edge between them. Strong smoothness is enforced in textureless areas, *e.g.*, between planes $\mathbf{D}_2$ and $\mathbf{D}_3$. The term $\mathrm{E_{smo}}$ enforces smoothness purely based on the segment labels, regardless of the distribution of pixel weights over different planes.

Effect on optical flow estimation Since any defocus incurs frequency loss, flow estimation is optimal when no aliasing occurs and when both blur radii $r_1 = r_2 = 0$. But DFD is not possible then. Theoretical analysis of the optimal r_1, r_2 thus requires trading off loss in depth accuracy against loss in flow accuracy. Ultimately, the optimal trade-off will be application specific. In principle, both flow and depth can be estimated well when r_1, r_2 are small, since DFD reliability depends on high frequencies too [23].

4. Global DFD

We now turn to the problem of representing depth and flow in a way that allows propagation of noisy likelihoods over large and irregularly-shaped regions.

Spline-based scene representation We represent the scene using four geometric quantities: (1) a set of N control points distributed over the image plane (Figure 7(a)); (2) a 3D plane associated with each control point that assigns a unique depth to every pixel; (3) a 2D affine transformation associated with each control point that assigns a unique flow vector to every pixel; and (4) an N-dimensional weight vector associated with every pixel $\mathbf{p}$ that describes the weight that individual control points have on the depth and flow at $\mathbf{p}$. Specifically, the depth $\mathrm{d}'_{\mathbf{p}}$ and flow $\mathbf{v}'_{\mathbf{p}}$ at pixel $\mathbf{p}$ is a weighted combination of the depths and flows assigned by each control point:

$$\mathrm{d}'_{\mathbf{p}} = {\mathbf{w_p}}^{\mathrm{T}}\mathbf{D}\mathbf{p}, \quad \mathbf{v}'_{\mathbf{p}} = \left[{\mathbf{w_p}}^{\mathrm{T}}\mathbf{U}\mathbf{p},\ {\mathbf{w_p}}^{\mathrm{T}}\mathbf{V}\mathbf{p}\right] \tag{8}$$

where matrices $\mathbf{D}, \mathbf{U}$ and $\mathbf{V}$ collect the 3D plane parameters and the affine flow parameters of the N control points. This representation models the scene as a collection of non-overlapping smooth segments whose spatial extent is determined by the weight vectors. The weight vectors allow for a very flexible scene model in which depth and flow discontinuities at segment boundaries are possible by ensuring that no two segments have pixels with non-zero weight for the same control point.

Constraints on weight vectors The weight vector $\mathbf{w_p}$ of each pixel is subject to two hard constraints. The first constraint enforces the convexity of weighted sums by requiring that a pixel's weights are non-negative and sum to one. The second constraint enforces consistency between spline parameters and the scene segmentation by requiring that control points and pixels belonging to different segments do not influence each other. Specifically, $\mathbf{w}_{\mathbf{p}n}$ is non-zero only if pixel $\mathbf{p}$ and the n-th control point belong to the same segment according to the segmentation $\mathcal{Z}$ of the scene.

Control points The N control points depend on image appearance and are computed in a pre-processing stage. Briefly, like many appearance-based segmentation techniques [14, 17], we associate a high-dimensional "feature vector" $\mathbf{f_p}$ to every pixel $\mathbf{p}$. Its 35 dimensions include 2D

image location, color in Lab space, and Laplacian eigenvectors [2]. The initial control points are computed by applying k-means clustering to these vectors and setting the number of clusters to N. The first two columns of $\mathbf{C}$ hold the image coordinates of the control points whereas the other columns can be thought of as describing image appearance in the control points' neighborhood. See [27] for full details.

The top-down likelihood $\mathrm{E}_{\mathbf{prior}}$ We define the top-down likelihood to be a sum of energy terms that generalize Yamaguchi *et al.*'s objective function [33]. A key novelty here is the inclusion of two spline-specific terms that are essential for handling very sparse data, and for obtaining stable inferences from a scene model as flexible as a spline. In particular, we combine five energy terms, two of which are novel ($\mathrm{E}_{\mathrm{ent}}, \mathrm{E}_{\mathrm{flat}}$) and three of which have been used before ($\mathrm{E}_{\mathrm{smo}}, \mathrm{E}_{\mathrm{bnd}}, \mathrm{E}_{\mathrm{img}}$):

$$\begin{aligned}\mathrm{E}_{\mathrm{prior}}(\mathcal{M}, \mathcal{Z}) \;&=\; \mathrm{E}_{\mathrm{ent}}(\mathbf{w}) + \lambda_f \mathrm{E}_{\mathrm{flat}}(\mathbf{D}, \mathbf{U}, \mathbf{V}, \mathbf{w}) \\ &+ \lambda_{\mathrm{s}} \mathrm{E}_{\mathrm{smo}}(\mathbf{D}, \mathbf{U}, \mathbf{V}, \mathbf{w}, \mathbf{t}) + \lambda_{\mathrm{i}} \mathrm{E}_{\mathrm{img}}(\mathbf{C}, \mathbf{w}) + \lambda_{\mathrm{b}} \mathrm{E}_{\mathrm{bnd}}(\mathrm{s}) \;\; . \end{aligned} \tag{9}$$

The negative entropy prior $\mathrm{E}_{\mathbf{ent}}$ encourages uniformity in the weights of individual pixels:

$$\mathrm{E}_{\mathrm{ent}}(\mathbf{w}) = \sum_{\mathbf{p}} \sum_{n} \mathbf{w}_{\mathbf{p}n} \log \mathbf{w}_{\mathbf{p}n} \;\; . \tag{10}$$

The inner sum of Eq. (10) can be thought of as measuring the negative entropy of pixel $\mathbf{p}$'s weight vector. As such, it reaches its minimum value of $-\log(N)$ when the weights in $\mathbf{w}_{\mathbf{p}}$ are distributed evenly among the N control points. Note that since the weights of each pixel are computed deterministically, this term does not measure uncertainty.

The flatness prior $\mathrm{E}_{\mathbf{flat}}$ encourages smoothness in image regions with slowly-varying weight vectors:

$$\mathrm{E}_{\mathrm{flat}}(\mathbf{D}, \mathbf{U}, \mathbf{V}, \mathbf{w}) = \sum_{\mathbf{p}} \sum_{m,n} \mathbf{w}_{\mathbf{p}m} \; \mathbf{w}_{\mathbf{p}n} \; \psi_{\mathbf{p}mn}(\mathbf{D}, \mathbf{U}, \mathbf{V}) \tag{11}$$

where the function $\psi_{\mathbf{p}mn}(\mathbf{D}, \mathbf{U}, \mathbf{V})$ measures the disagreement between planes $\mathbf{D}_m$ and $\mathbf{D}_n$ at pixel $\mathbf{p}$:

$$\psi_{\mathbf{p}mn}(\mathbf{D}, \mathbf{U}, \mathbf{V}) = \frac{|\mathbf{D}_m\mathbf{p} - \mathbf{D}_n\mathbf{p}|^2}{2\sigma_d^2} + \frac{|\mathbf{U}_m\mathbf{p} - \mathbf{U}_n\mathbf{p}|^2}{2\sigma_{\mathbf{v}}^2} + \frac{|\mathbf{V}_m\mathbf{p} - \mathbf{V}_n\mathbf{p}|^2}{2\sigma_{\mathbf{v}}^2} \;\; . \tag{12}$$

Intuitively, $\mathrm{E}_{\mathrm{flat}}$ penalizes weight vectors $\mathbf{w}_{\mathbf{p}}$ that assign non-zero weights to control points whose depth and flow predictions are inconsistent at $\mathbf{p}$. As such, it prevents smoothing across image boundaries. This behavior suggests a relation to the consistency term of the hierarchical consensus framework [7]. That optimization architecture, however, is very different from ours, as it handles discontinuities only at the patch level and does not infer a global segmentation model.

Previously-proposed terms: $\mathrm{E}_{\mathbf{smo}}, \mathrm{E}_{\mathbf{img}}$ and $\mathrm{E}_{\mathbf{bnd}}$ These already appear in the literature [33]. The segment-specific smoothing term $\mathrm{E}_{\mathrm{smo}}$ is similar to the smoothing energy of the piecewise-planar model in [33]. It is defined as

$$\mathrm{E}_{\mathrm{smo}}(\mathbf{D}, \mathbf{U}, \mathbf{V}, \mathbf{w}, \mathbf{t}) = \sum_{\mathbf{p}} \sum_{m,n} [\mathbf{w}_{\mathbf{p}m} + \mathbf{w}_{\mathbf{p}n}] \cdot [\psi_{\mathbf{p}mn}(\mathbf{D}, \mathbf{U}, \mathbf{V}) \delta_{mn} + (1 - \delta_{mn}) \tau_{\mathrm{s}}] \tag{13}$$

Algorithm 1: Global DFD

input : initial control points $\mathbf{C}$, feature map $\mathbf{f}$, patch likelihoods Q
output: patch-based depth and flow $\mathcal{P} = (\mathrm{d}, \mathbf{v})$, spline parameters $\mathcal{M} = (\mathbf{C}, \mathbf{D}, \mathbf{U}, \mathbf{V}, \mathbf{w})$, scene segmentation $\mathcal{Z} = (\mathrm{s}, \mathbf{t}, \mathbf{O})$, pixel-based depth and flow $\mathcal{S}(\mathcal{M}) = (\mathrm{d}', \mathbf{v}')$

1 initialize $\mathcal{S} = (\mathbf{0}, \mathbf{0}), \mathbf{D} = \mathbf{U} = \mathbf{V} = 0, \; \mathbf{t} = \mathbf{0}$
2 **repeat**
3 update s and $\mathbf{w}$ jointly by solving MRF
4 update $\mathbf{O}$ by thresholding
5 update $\mathcal{P}$, $\mathbf{D}$, $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{t}$ jointly by solving IRLS
6 update $\mathbf{C}_n = \sum_{\mathbf{p}} (\mathbf{w}_{\mathbf{p}n} \mathbf{f}_{\mathbf{p}}) / \sum_{\mathbf{p}} \mathbf{w}_{\mathbf{p}n}$
7 **until** *convergence*
8 compute $\mathcal{S}(\mathcal{M})$ from spline parameters $\mathcal{M}$ using Eq. (8).

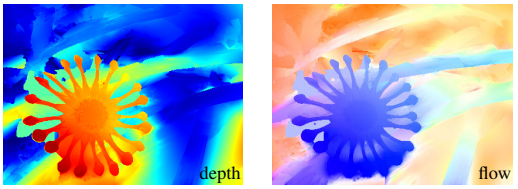


Figure 8: Global depth and flow when E_{flat} is disabled (*i.e.*, $\lambda_f = 0$ in the energy function of Eq. (9)). Compare these results to the far superior results in Figure 1 (columns 5,6).

where the binary variable δ_{mn} is one if and only if (1) control points m and n belong to the same segment (*i.e.*, $\mathbf{t}_m = \mathbf{t}_n$) and (2) they are not too far apart (*i.e.*, less than 10% of the image diagonal). τ_{s} penalizes over-segmentation.

The remaining two terms encourage image coherence and penalize discontinuities:

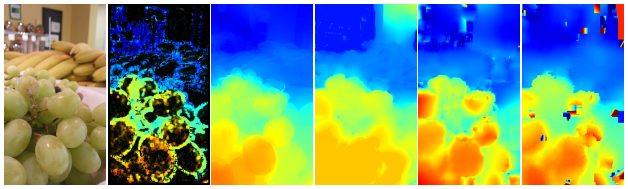
$$\mathrm{E}_{\mathrm{img}}(\mathbf{C}, \mathbf{w}) = \textstyle\sum_{\mathbf{p}} \sum_{n} \mathbf{w}_{\mathbf{p}n} |\mathbf{f}_{\mathbf{p}} - \mathbf{C}_n|^2 \tag{14}$$

$$\mathrm{E}_{\mathrm{bnd}}(\mathrm{s}) = \textstyle\sum_{\mathbf{p},\mathbf{q} \in \mathcal{N}_8} (1 - \delta(\mathrm{s}_{\mathbf{p}} - \mathrm{s}_{\mathbf{q}})) \tag{15}$$

where $\delta(\,)$ is Dirac's Delta function. In particular, $\mathrm{E}_{\mathrm{img}}$ acts as a soft segmentation constraint that forces the weights in $\mathbf{w}_{\mathbf{p}}$ to be non-zero only for control points whose feature vector is similar to $\mathbf{p}$'s. See [33] for a detailed discussion.

Optimization We solve the optimization problem in Eq. (1) using a standard block-coordinate descent scheme (Algorithm 1). Notice that both step 3 and step 5 involve *global* optimization. This allows our method to propagate local information potentially across the entire image. *We always use a trivial initialization*: all our results were obtained by setting all variables to zero in step 1. See [27] for more details on the algorithm.

Ablation study The local prior L in Eq. (6) couples the patch-based and pixel-based depth maps and flow fields. Disabling this term causes the optimization to trivially return the patch-based depth map and flow field. This is because the pixel-based variables are not subject to any data constraints. Disabling both terms $\mathrm{E}_{\mathrm{ent}}$ and $\mathrm{E}_{\mathrm{flat}}$ in Eq. (9) makes our prior identical to the SPS prior [33, 34]. As we show in Section 5, enabling both terms outperforms the SPS prior. Disabling only term $\mathrm{E}_{\mathrm{ent}}$ produces results identical to SPS. This is because $\mathrm{E}_{\mathrm{ent}}$ is the only term encouraging pixels to be owned by different control points. Disabling only term $\mathrm{E}_{\mathrm{flat}}$ produces over-smoothed results (Figure 8).



input image 1 LDFD GDFD DFF [26] HCF [7] SPS [34]

Figure 9: Qualitative comparison with related work on the Samsung images. Our method performs much better than the DFF method in [26] which requires 30 frames. It also outperforms two semi-dense methods that are applied to the results of LDFD. (See [27] for flow results and additional scenes.)

5. Results

We have tested our approach on a variety scenes and imaging conditions that are very challenging for DFD. Table 1 summarizes the experimental conditions for the results shown in the paper. They include scenes with significant deformation, large depth variations, and a fairly diverse set of surface textures (or lack thereof). More results can be found in [27, 28].

Parameters Our results for all cameras and all scenes were obtained with *exactly the same parameters*—except for the noise level which was set according to camera ISO. We used $\tau_i = 30$, $\tau_o = 0.01$, $\tau_s = 0.01$, $\lambda_f = 2$, $\lambda_\mathrm{s} = 0.00001$, $\lambda_\mathrm{i} = 1000$ and $\lambda_\mathrm{b} = 5$.

Description of datasets Our SLR camera (a Canon 7D with 50mm f1.2L lens) and one of the cellphone cameras (the Nexus5 with F2.4 lens) output high-resolution RAW images. These had dimensions 5184×3456 and 2464×3280, respectively, and were captured by us under active focus control. Data for the second cellphone camera (Samsung S3) were provided by the authors of [26] and used here for comparison purposes. Unlike the other two, they consist of JPEG focal stacks with 23 to 33 JPEG images of size 360×640. We use just two images from these stacks for DFD, in accordance with the tiny blur condition. The exact same tiny blur condition was used for all datasets despite their differences in image resolution.

Organization of results We show two sets of results per dataset: (1) the depth and flow maps computed by Local DFD (LDFD) and (2) the maps computed by Global DFD (GDFD). The former are the maximum-likelihood values, $\mathrm{d}_\mathbf{p}^*$ and $\mathbf{v}_\mathbf{p}^*$, in Eq. (5); the latter are obtained by optimizing Eq. (1). This optimization turns the sparse and noisier LDFD results into high-quality dense maps.

Qualitative results on real scenes Figure 10 shows results on several complex scenes. Some of these scenes exhibit very significant deformation, as can be seen from the flow estimates in the last column. Despite this—and with just two frames to work with—LDFD already produces results of high quality. GDFD inpaints and regularizes the LDFD data while preserving sharp boundaries. Moreover, the segments it recovers can be curved rather than fronto-parallel.

Comparison to Depth from Focus (DFF) [26] Figure 9 compares the results of our *two-image* method to the results of a recent DFF method that uses all 23 *to* 41 *images* in the focal stack. Observe that both our LDFD and GDFD results are more detailed and have fewer outliers than [26].

GDFD versus semi-dense methods [7, 34] Figure 9 also compares two approaches: (1) applying GDFD to the Samsung dataset and (2) applying two recent semi-dense methods to the sparse LDFD results on that dataset in order to compute dense depth and flow. This comparison shows that GDFD is more robust, produces sharper boundaries and adheres more closely to scene appearance.

Quantitative results on synthetic data We simulate synthetic data from the Middlebury stereo 2006 dataset (full-size version) [13]. Since occluded pixels have no disparity in this dataset's disparity maps, we first inpaint the missing disparities using a recent depth-inpainting technique [11]. We then convert the disparity map to a map of blur kernels by linearly mapping the disparity range $[0, 255]$ to the range $[-4, 2]$ for the blur kernel radius r_1. We obtain the first input image via a spatially-varying convolution of the dataset's left image and this blur kernel map. To simulate a differently-blurred second input image, we repeat the process after setting $r_2 = r_1 + 2$ at the corresponding pixel in the blur kernel map. Finally, we add Gaussian noise of variance 10^{-4} to both images.

As shown in Table 2, LDFD yields low error for all scenes. The proportion of confident patches depends on scene content: when confident patches are sparse, GDFD estimates a dense depth and flow map whereas the semi-dense methods perform worse due to the sparsity of the LDFD results. When confident patches are more dense, all methods work well but GDFD produces the smallest depth error overall.

Running time We run Matlab code on a server with 128GB of RAM and two 8 core 2.6Ghz Xeon processors. On the Nexus images, LDFD takes about fifteen minutes, control point initialization takes five, and GDFD takes fourty minutes to one hour.

6. Concluding Remarks

In this work we have shown that despite the problem's apparent hardness, two-frame DFD offers a promising way to recover depth and flow from minimal visual input in everyday settings. As with other "minimal" methods [12], we believe that the technique's parsimony is its greatest strength—it opens the door to generalizations that involve richer visual input (*e.g.*, video streams, focal stacks, aperture stacks) and new types of 3D analysis (*e.g.*, combining two-frame DFD with structure from motion and/or stereo).

Acknowledgements The support of Adobe Inc. and of the Natural Sciences and Engineering Research Council of Canada (NSERC) under the RPGIN and SPG programs are gratefully acknowledged.

	avg		Aloe		Baby1		Cloth1		Flowerpots		Rock1		Wood1	
	depth	flow	depth	flow	depth	flow	depth	flow	depth	flow	depth	flow	depth	flow
LDFD density (ours)	46.1%		61.8%		36.2%		83.6%		13.2%		71.9%		46.1%	
LDFD error (ours)	0.18	0.66	0.12	0.53	0.18	0.69	0.13	0.31	0.36	1.63	0.23	0.96	0.25	0.74
LDFD+GDFD errors (ours)	**0.19**	**2.22**	**0.17**	**3.70**	**0.19**	**1.65**	**0.11**	1.25	**0.23**	**2.24**	**0.21**	1.62	**0.21**	**2.87**
LDFD+SPS [34] errors	0.34	3.75	0.27	4.20	0.29	2.79	0.17	**0.98**	0.64	8.18	0.26	**1.60**	0.38	4.75
LDFD+HCF[7] errors	0.36	7.62	0.24	4.47	0.41	2.18	0.29	7.73	0.53	17.09	0.37	8.26	0.31	6.03

Table 2: Quantitative evaluation on synthetic data with groundtruth. We evaluate the LDFD results by measuring the proportion of locally confident patches (LDFD density) and mean end-point errors in the depth map and flow field. The error in depth is measured as the error in defocus blur radius of the first image. The sparse LDFD results are used as input to both our Global DFD method as well as two recent semi-dense methods (SPS [34] and HCF [34]). We compare the error of GDFD results for each scene in depth and flow, and show the smallest error in bold. Global DFD performs well in all cases.

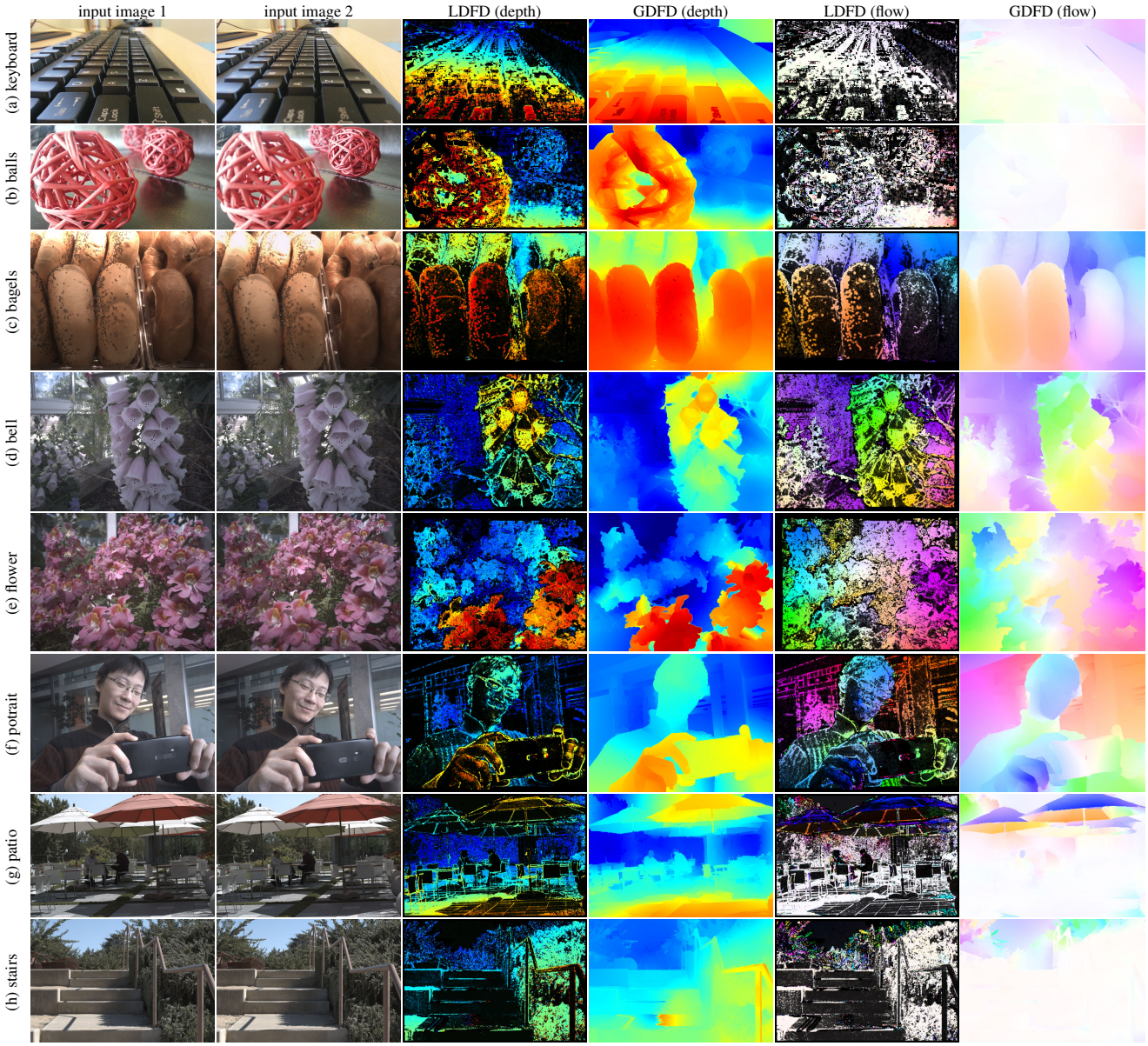


Figure 10: LDFD and GDFD results on eight scenes captured by three cameras. See Table 1 for a summary of the scenes, cameras and image conditions. Zoom in to appreciate the fine structures in the depth and flow maps our method computes.

References

[1] E. Alexander, G. Q., K. S. J., and Z. T. Focal flow: Measuring distance and velocity with defocus differential motion. In *Proc. ECCV*, 2016.

[2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proc. IEEE CVPR*, 2014.

[3] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proc. IEEE ICCV*, 2015.

[4] S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *Int. J. Computer Vision*, 56(3):221–255, 2004.

[5] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Proc. IEEE ICCV*, 1999.

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE T-PAMI*, 23(11):1222–1239, 2001.

[7] A. Chakrabarti, Y. Xiong, S. J. Gortler, and T. Zickler. Low-level vision by consensus in a spatial hierarchy of regions. In *Proc. IEEE CVPR*, pages 4009–4017, 2015.

[8] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *Proc. ECCV*, pages 834–849, 2014.

[9] P. Favaro. Recovering thin structures via nonlocal-means regularization with application to depth from defocus. In *Proc. IEEE CVPR*, 2010.

[10] P. Favaro, S. Soatto, M. Burger, and S. J. Osher. Shape from defocus via diffusion. *IEEE T-PAMI*, 30(3):518–531, 2008.

[11] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *Proc. IEEE ICCV*, 2013.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[13] H. Hirschmuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *Proc. IEEE CVPR*, 2007.

[14] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *Proc. ECCV*, 2014.

[15] V. Lempitsky, S. Roth, and C. Rother. FusionFlow: Discrete-continuous optimization for optical flow estimation. In *Proc. IEEE CVPR*, 2008.

[16] F. Li, J. Sun, J. Wang, and J. Yu. Dual focus stereo imaging. In *SPIE Electronic Imaging*, 2010.

[17] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries. In *Proc. ECCV*, 2016.

[18] M. Moeller, M. Benning, C. Schonlieb, and D. Cremers. Variational Depth From Focus Reconstruction. *IEEE TIP*, 24(12):5369–5378, 2015.

[19] A. P. Pentland. A New Sense for Depth of Field. *IEEE T-PAMI*, (4):523–531, 1987.

[20] T. Portz, L. Zhang, and H. Jiang. Optical flow in the presence of spatially-varying motion blur. In *Proc. IEEE CVPR*, 2012.

[21] A. N. Rajagopalan and S. Chaudhuri. A variational approach to recovering depth from defocused images. *IEEE T-PAMI*, 19(10):1158–1164, Oct. 1997.

[22] A. N. Rajagopalan and S. Chaudhuri. Optimal selection of camera parameters for recovery of depth from defocused images. In *Proc. IEEE CVPR*, 1997.

[23] Y. Schechner and N. Kiryati. The optimal axial interval in estimating depth from defocus. In *Proc. IEEE CVPR*, 1999.

[24] N. Shroff, A. Veeraraghavan, Y. Taguchi, O. Tuzel, A. Agrawal, and R. Chellappa. Variable focus video: Reconstructing depth and video for dynamic scenes. In *Proc. IEEE ICCV*, 2012.

[25] F. Sroubek and P. Milanfar. Robust multichannel blind deconvolution via fast alternating minimization. *IEEE Trans. Image Processing*, 21(4):1687–1700, 2012.

[26] S. Suwajanakorn, C. Hernández, and S. M. Seitz. Depth from focus with your mobile phone. In *Proc. IEEE CVPR*, pages 3497–3506, 2015.

[27] H. Tang, S. Cohen, B. Price, S. Schiller, and K. N. Kutulakos. Depth from defocus in the wild: supplementary material. 2017.

[28] *Depth from defocus in the wild: Project webpage.* [Online] (2017). Available at http://www.dgp.toronto.edu/WildDFD.

[29] *How does the L16 work?* [Online] (2017). Available at https://light.co/technology.

[30] C. Vogel, K. Schindler, and S. Roth. Piecewise Rigid Scene Flow. In *Proc. IEEE ICCV*, pages 1377–1384, 2013.

[31] M. Watanabe and S. K. Nayar. Rational filters for passive depth from defocus. *Int. J. Computer Vision*, 27(3):203–225, 1998.

[32] T. Xian and M. Subbarao. Depth-from-defocus: blur equalization technique. *Proc. SPIE*, 6382:1–10, 2006.

[33] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation. In *Proc. ECCV*, pages 756–771, 2014.

[34] J. Yao, M. Boben, S. Fidler, and R. Urtasun. Real-Time Coarse-to-fine Topologically Preserving Segmentation. In *Proc. IEEE CVPR*, 2015.

[35] F. Yu and D. Gallup. 3D Reconstruction from Accidental Motion. In *Proc. IEEE CVPR*, 2014.

[36] C. Zhou, S. Lin, and S. K. Nayar. Coded Aperture Pairs for Depth from Defocus and Defocus Deblurring. *Int. J. Computer Vision*, 93(1):53–72, Dec. 2010.