Focal Flow

Velocity and Depth from Differential Defocus through Motion

Emma Alexander · Qi Guo · Sanjeev Koppal · Steven J. Gortler · Todd Zickler

Abstract We present the focal flow sensor. It is an unactuated, monocular camera that simultaneously exploits defocus and differential motion to measure a depth map and a 3D scene velocity field. It does this using an optical-flow-like, per-pixel linear constraint that relates image derivatives to depth and velocity. We derive this constraint, prove its invariance to scene texture, and prove that it is exactly satisfied only when the sensor's blur kernels are Gaussian. We analyze the inherent sensitivity of the focal flow cue, and we build and test a prototype. Experiments produce useful depth and velocity information for a broader set of aperture configurations, including a simple lens with a pillbox aperture.

Keywords depth · optical flow · defocus · coded aperture · ego-motion · computational sensing

Computational sensors reduce the data processing burden of visual sensing tasks by physically manipulating light on its path to a photosensor. They analyze scenes using vision algorithms, optics, and post-capture computation that are jointly designed for a specific task or environment. By optimizing which light rays are sampled, and by moving some of the computation from electrical hardware into the optical domain, computational sensors promise to extend task-specific artificial vision to new extremes in size, autonomy, and power consumption [24,14,5,9,15].

This paper presents a computational sensor for simultaneous depth and 3D scene velocity. It is called a *focal flow sensor*. It is passive and monocular, and it measures depth and velocity using a per-pixel linear constraint composed of spatial and temporal image derivatives. The sensor simultaneously exploits defocus and differential motion, and its underlying principle is depicted in Figure 1. This figure shows the one-dimensional image values that would be measured from a front-parallel, Lambertian scene patch with a sinusoidal texture pattern, as it moves relative to a sensor. If the sensor is a pinhole camera, the patch is always in focus, and the images captured over time are variously stretched and shifted versions of the patch's texture pattern (Figure 1A). The rates of stretching and shifting together resolve the direction of motion and time to contact (e.g., using [11]), but they are not sufficient to explicitly measure depth or velocity. The focal flow sensor is a real-aperture camera with a finite depth of field, so in addition to stretching and shifting, its images exhibit changes in contrast due to defocus (Figure 1B). This additional piece of information resolves depth and velocity explicitly.

Our main contribution is the derivation of a per-pixel linear equation,

$$\boxed{\begin{bmatrix} I_x & I_y & (xI_x + yI_y) & \nabla^2 I \end{bmatrix} \cdot \mathbf{v} + I_t = 0,}$$

that relates spatial and temporal image derivatives to depth and 3D scene velocity, and that is valid *for any generic scene texture*. Over an image patch, depth and velocity are recovered simply by computing partial derivatives in time (I_t) and space (I_x, I_y, $\nabla^2 I$), solving a 4×4 linear system for vector $\mathbf{v} \in \mathbb{R}^4$, and then evaluating analytic expressions for depth $Z(\mathbf{v})$ and 3D velocity $(\dot{X}, \dot{Y}, \dot{Z})(\mathbf{v})$ determined by the physical characteristics of the calibrated sensor.

The focal flow cue is distinct from conventional passive depth cues like stereo and depth from defocus because it directly measures 3D velocity in addition to

E. Alexander
E-mail: ealexander@fas.harvard.edu

E. Alexander · Qi Guo · S.J. Gortler · T. Zickler
Harvard University

Sanjeev Koppal
University of Florida

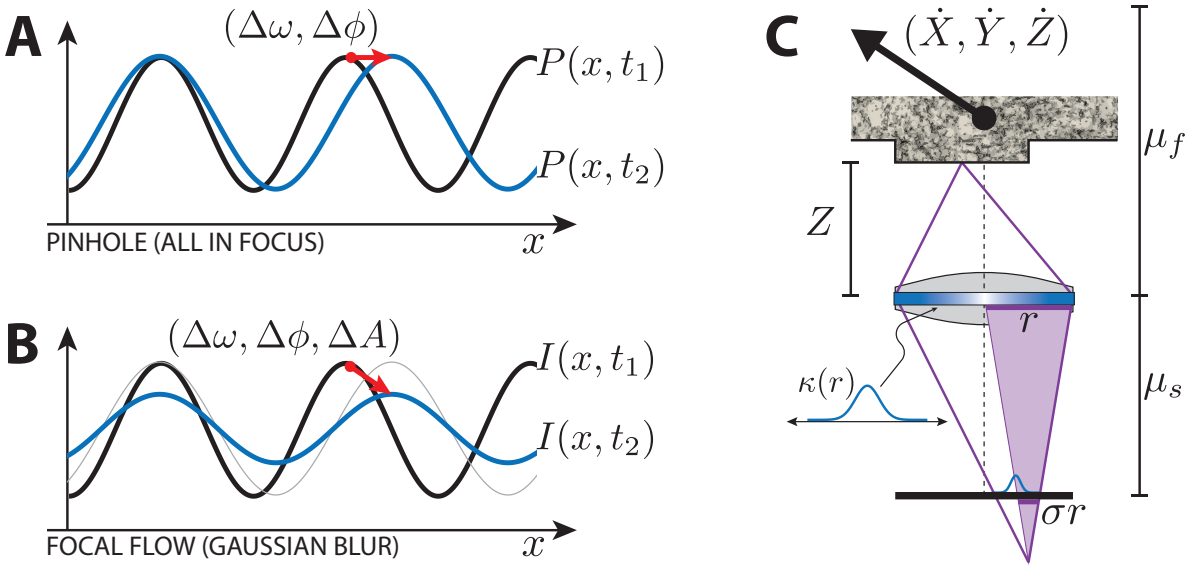


Fig. 1 The focal flow principle. A: When a 1D pinhole camera observes a world plane with sinusoidal texture, the image is also a sinusoid (black curve). Motion between the camera and scene causes the sinusoidal image to change in frequency and phase (blue curve), and these two pieces of information reveal time to contact and direction of motion. **B**: When a finite-aperture camera images a similar moving scene, the motion additionally induces a change in image amplitude, because the scene moves in or out of focus. This third piece of information resolves depth and scene velocity. **C**: We show that, with an ideal thin lens and Gaussian blur $\kappa(r)$, depth and 3D velocity can be measured through a simple, per-pixel linear constraint, similar optical flow. The constraint applies to any generic scene texture.

depth. Importantly, it does not require inferences about disparity or blur; instead, it provides per-pixel depth in closed form, using a relatively small number of multiply and add operations. The focal flow sensor might therefore be useful for applications, such as micro-robotics [5], that involve motion and that require visual sensing with low power consumption and small form factors.

The heart of this paper is devoted to proving that this linear constraint is invariant to scene texture, that it exists analytically whenever the optical system's point spread functions are Gaussian, and that no other class of point spread functions—be they discs, binary codes, or continuous functions—provides the same analytic capabilities. We also analyze the first order sensitivity of the focal flow sensor and compare it to those of stereo and depth from defocus.

In the second part of the paper, we formulate the patch-based computation of depth and velocity as a feed-forward computational tree, and we develop techniques for end-to-end calibration that maximize depth accuracy by simultaneously optimizing the discrete spatial derivative filters and the precise values of the sensor's relevant optical dimensions. Experimentally, we find this allows effective depth estimation using point spread functions that deviate substantially from Gaussians, including those that arise from a simple, bare lens. We demonstrate a working prototype that can measure depth within ±6mm over a range of 20cm using a one-inch lens.

1 Related Work

This paper is an extended version of [1]. It presents a more general characterization of the allowable point spread functions by dropping the restriction of radially symmetric optics. This slightly widens the family of point spread functions that analytically allow a focal flow constraint, and it greatly increases the set of those that do not. This paper also introduces a computational model that supports end-to-end calibration, which empirically provides a significant improvement in performance.

Motion & Linear Constraints. Differential optical flow, which assumes that all images are in focus, is computable from a linear system of equations in a window [13]. A closely related linear system resolves time to contact [11,16]. The focal flow equation has a similar linear form, but it incorporates defocus blur and provides additional scene information in the form of depth and 3D velocity. Unlike previous work on time to contact [12], our focal flow analysis is restricted to front-parallel scene patches, though experimental results suggest that useful depth can be obtained for some slanted planes as well (see Figure 6).

Defocus. When many images are collected under a variety of calibrated camera settings, a search for the most-in-focus image will yield depth [10]. This approach is called depth from focus, and it is reliable but expensive in terms of time and images captured. When restricted to a few images, none of which are guaranteed to be in focus, a depth from defocus algorithm must be used [23]. This method is more difficult because the underlying texture is unknown: we cannot tell if we're seeing a blurry picture of an oil painting or the sharp image of a watercolor, and without accurate image priors both solutions are equally valid. To reduce ambiguity, most depth from defocus techniques require at least two exposures with substantially different blur kernels, controlled by internal camera actuation that changes the focal length or aperture diaphragm to manipulate the blur kernel [23,32,25,37]. The speed, accuracy, and complexity of recovering depth depends on the blur kernels and the statistical image model that is used for inference. Depth performance improves when well-designed binary attenuation patterns are included in the aperture plane [39,17,40], and with appropriate inference, binary codes can even provide useful depth from a single exposure [18,36,3].

Focal flow is similar to depth from defocus in that it relies on focus changes over a small set of images to reveal depth, and that it requires a specific blur kernel. However, the implied hardware is different: unlike multi-shot depth from defocus, our sensor does not require internal actuation, and unlike binary aperture codes, it employs a continuous filter.

Differential defocus with Gaussian blur has been previously considered under changing lens and aperture conditions [31,32,6]. We build on this work by proving the uniqueness of the Gaussian filter, and by exploiting differential motion to avoid camera actuation.

Cue Combination. Our use of relative motion between scene and sensor means that in many settings, such as robotics or motion-based interfaces, this cue comes without an additional power cost. Previous efforts to combine camera/scene motion and defocus cues [20,7,19,29,22,30,34] require intensive computations, though they often account for motion blur, which we ignore. Even when motion is known, equivalent to combining defocus with stereo, measuring depth still requires searching over a discrete set of depth estimates [26,35]. The simplicity of focal flow provides an advantage in efficiency.

2 The Focal Flow Constraint

In differential optical flow, a pinhole camera views a Lambertian object with a temporally constant albedo pattern, here called texture and denoted $T : \mathbb{R}^2 \to [0,\infty)$. For now the texture is assumed to be differentiable, but this requirement will be relaxed to local integrability for the derivation of focal flow. For front-parallel planar objects, located at a time-varying offset (X,Y) and depth Z from the pinhole, the camera captures an all-in-focus image that varies in time t and pixel location (x,y) over a bounded patch S on a sensor located an axial distance μ_s from the pinhole. The intensity of this image $P : S \times \mathbb{R} \to [0,1]$ is a magnified and translated version of the texture, scaled by an exposure-dependent constant η:

$$P(x,y,t) = \eta\, T\left(\frac{Z(t)}{-\mu_s}x - X(t),\ \frac{Z(t)}{-\mu_s}y - Y(t)\right). \quad (1)$$

It is well known that the ratios of the spatial and temporal derivatives of this image are independent of texture, and so can reveal information about the scene. A familiar formulation [13] provides optical flow $(\dot{x},\dot{y})$ from image derivatives:

$$0 = \begin{bmatrix} P_x & P_y \end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} + P_t, \quad (2)$$

while following [11] to split the translation and magnification terms:

$$0 = \begin{bmatrix} P_x & P_y & (xP_x + yP_y)\end{bmatrix}\mathbf{u} + P_t, \quad (3)$$

$$\mathbf{u} = [u_1,u_2,u_3]^T = \left[-\frac{\dot{X}\mu_s}{Z}, -\frac{\dot{Y}\mu_s}{Z}, -\frac{\dot{Z}}{Z}\right]^T, \quad (4)$$

provides texture-independent time to contact and direction of motion (bearing):

$$\frac{Z}{\dot{Z}} = \frac{-1}{u_3}, \quad (5)$$

$$\left(\frac{\dot{X}}{\dot{Z}}, \frac{\dot{Y}}{\dot{Z}}\right) = \left(\frac{u_1}{\mu_s u_3}, \frac{u_2}{\mu_s u_3}\right). \quad (6)$$

For focal flow, we replace the pinhole camera with a finite-aperture camera having an ideal thin lens and an attenuating filter in the aperture plane. We represent the spatial transmittance profile of the filter with the function $\kappa : \mathbb{R}^2 \to [0,1]$. We do not require smoothness, which allows for pillboxes and binary codes as well as continuous filters. For a front-parallel world plane at depth Z, the filter induces a blur kernel k on the image that is a "stretched" version of the aperture filter:

$$k(x,y;Z) = \frac{1}{\sigma^2(Z)}\kappa\left(\frac{x}{\sigma(Z)}, \frac{y}{\sigma(Z)}\right), \quad (7)$$

where the magnification factor σ, illustrated in Figure 1C, is determined by object depth, sensor distance, and in-focus depth μ_f via the thin lens model:

$$\sigma(Z) = \left(\frac{1}{Z} - \frac{1}{\mu_f}\right)\mu_s. \tag{8}$$

Denoting by $*$ a convolution in x and y, we can write the blurred image I as

$$I(x,y,t) = k\left(x,y;Z(t)\right) * P(x,y,t). \tag{9}$$

Unlike the pinhole image P, the ratios of the spatial and temporal derivatives of this defocus-blurred image I depend on texture. This is because the constant brightness constraint does not hold under defocus: pixel intensity changes both as image features move and also as patch contrast is reduced away from the focal plane. This difference, illustrated in Figure 1, implies that any finite-aperture system for measuring optical flow will suffer a systematic error from defocus. Mathematically, this appears as an additive residual term on the time derivative, as shown in the following proposition.

Proposition. *For an ideal thin lens camera and front-parallel planar scene, denoting by k_x and k_y the partial, distributional derivatives of k,*

$$I_t = k_t * P + k * P_t \tag{10}$$
$$= -u_1 I_x - u_2 I_y - u_3(xI_x + yI_y) - R, \tag{11}$$
$$R = \frac{\dot{Z}}{Z - \mu_f}(2k + xk_x + yk_y) * P. \tag{12}$$

The time-varying residual image $R(x,y,t)$ changes with depth, velocity, and camera design. It is troublesome because it also depends on the pinhole image P, which is not directly measured. Only the blurred image $I = k*P$ is available. This means that for almost all aperture filters, there is no way to express R using scene geometry and image information alone—it is inherently texture-dependent.

However, we observe that for a very specific aperture filter, this source of error can actually be transformed into a usable signal that resolves both depth and 3D velocity. For this to happen, the image must be processed with a particular operation that, in combination with the filter, allows the decomposition of residual image R into a depth/velocity factor (analogous to u_1) and an accessible measurement (analogous to I_x). To formally identify such a filter and image operator, we seek triples (M, κ, v) of shift-invariant linear image operators M, aperture filters κ, and scalar scene factors v that satisfy, for any front-parallel planar scene,

$$v(t)\ M[I](x,y) = R(x,y,t). \tag{13}$$

We prove in the following theorem that there exists a unique family of such triples, comprising Gaussian aperture filters and Laplacian image measurements. This leads directly to a simple sensor and algorithm that we prototype and evaluate in Section 4.

Theorem. *Let k be induced by some $\kappa : \mathbb{R}^2 \to [0,1]$ with $\kappa(x,y)$, $x\kappa(x,y)$, and $y\kappa(x,y)$ Lebesgue integrable and κ not identically zero. For $v \in \mathbb{R}$ and translation-invariant linear spatial operator M with finite support,*

$$v\ M\left[k * P\right] = R(k,P) \tag{14}$$

for all compactly supported P, if and only if there are constants $a \in \mathbb{R}^+$, $b, \Sigma \in \{\mathbb{R} - 0\}$ and a real symmetric positive definite matrix $\boldsymbol{\Sigma}$ such that

$$\kappa = a\ e^{-\frac{\mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x}}{4\Sigma^2 |\boldsymbol{\Sigma}|}}, \tag{15}$$
$$M = b\ \nabla^2_{\boldsymbol{\Sigma}} = \ b\ \partial_{\mathbf{x}}^T \boldsymbol{\Sigma}^{-1} \partial_{\mathbf{x}}. \tag{16}$$

This theorem states that, when the filter κ is a Gaussian with covariance matrix $\boldsymbol{\Sigma}$, the residual R is proportional to the image Laplacian of inverse covariance, $M[I] \propto \nabla^2_{\boldsymbol{\Sigma}} I = \boldsymbol{\Sigma}^{-1}_{11} I_{xx} + 2\boldsymbol{\Sigma}^{-1}_{12} I_{xy} + \boldsymbol{\Sigma}^{-1}_{22} I_{yy}$, which is directly observable from image information. Moreover, the Gaussian is the *only* aperture filter—out of a broad class of possibilities including pillboxes, binary codes, and smooth functions—that permits exact observation by a depth-blind, translation-invariant linear operator.

Combining the proposition and theorem leads immediately to a per-pixel linear constraint, analogous to those used in measuring optical flow or time to contact.

Corollary (Focal Flow Constraint). *For a camera with Gaussian point spread functions observing a front-parallel planar scene, the following constraint holds at each image pixel:*

$$0 = \begin{bmatrix} I_x & I_y & (xI_x + yI_y) & \nabla^2_{\boldsymbol{\Sigma}} I \end{bmatrix} \mathbf{v} + I_t,$$
$$\mathbf{v} = [u_1, u_2, u_3, v]^T$$
$$= -\left[\frac{\dot{X}\mu_s}{Z}, \frac{\dot{Y}\mu_s}{Z}, \frac{\dot{Z}}{Z}, \frac{\dot{Z}}{Z}\left(1 - \frac{\mu_f}{Z}\right)\left(\frac{\Sigma\mu_s}{\mu_f}\right)^2 2|\boldsymbol{\Sigma}|\right]^T. \tag{17}$$

Note that the boxed equation in the introduction is one member of this family of possible constraints, corresponding to the case where $\boldsymbol{\Sigma}$ is the identity matrix and the point spread functions of the camera are radially symmetric. For a calibrated camera, holding this constraint over a generic image patch yields a system of linear equations that can be solved for $\mathbf{v}$. In the presence of axial motion ($\dot{Z} \neq 0$) the new scalar factor v

provides enough additional information to directly recover complete depth and velocity:

$$Z = \frac{(2|\boldsymbol{\Sigma}|\Sigma^2\mu_s^2\mu_f)\ u_3}{(2|\boldsymbol{\Sigma}|\Sigma^2\mu_s^2)\ u_3 - (\mu_f^2)\ v}, \tag{18}$$

$$(\dot{X}, \dot{Y}, \dot{Z}) = -\left(Zu_1/\mu_s, Zu_2/\mu_s, Zu_3\right), \tag{19}$$

where components of $\mathbf{v}$ are measured from the image and all other parameters are known from camera calibration. Note that the corollary drops the assumption of a compactly-supported texture (see appendix A for details): the constraint holds exactly for any front-parallel textured plane.

This implies a simple patch-wise algorithm for measuring depth and velocity, about which we make a few notes. When an image patch is degenerate, meaning that the matrix having a row $[I_x, I_y, xI_x + yI_y, \nabla^2_{\boldsymbol{\Sigma}} I]$ for each of the patch's pixels is not full rank, partial scene information can often still be obtained. For example, a patch that contains a single-orientation texture and is subject to the classical aperture problem gives rise to ambiguities in the lateral velocity $(\dot{X}, \dot{Y})$, but depth Z and axial velocity $\dot{Z}$ can still be determined. Separately, in the case of zero axial motion ($\dot{Z} = 0$), there is no change in defocus and the depth signal is lost. Specifically, $u_3 = v = 0$, and the patch can only provide optical flow. Finally, note that unlike many depth from defocus methods, the combination of magnification and defocus changes in focal flow breaks the side-of-focal-plane ambiguity.

The following proofs draw heavily on the theory of distributions, for which we suggest [27] as a reference. A brief introduction to the relevant terms and properties can be found in appendix A. Intuition may be gained from appendix B, which contains alternate derivations for the focal flow constraint under the assumption of Gaussian blur.

2.1 Proofs

Proof (Proposition) We can rewrite the first term in equation (10) using the differential optical flow constraint, which expresses P_t in terms of spatial derivatives P_x and P_y:

$$k * P_t = k * \left(-\begin{bmatrix} P_x & P_y & (xP_x + yP_y) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \right). \tag{20}$$

The magnification term $xP_x + yP_y$ introduces a complication, because the image coordinates x and y cannot be pulled out of the convolution without introducing additional terms:

$$k * xP_x = x(k * P_x) - (xk * P_x). \tag{21}$$

Spatial derivatives can be applied to either term in a spatial convolution, so this new term takes the form

$$xk * P_x = (k + xk_x) * P, \tag{22}$$

and equation (20) can be rewritten in terms of image measurements with leftover P terms:

$$\begin{aligned} k * P_t = &- \begin{bmatrix} I_x & I_y & (xI_x + yI_y) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \\ &+ u_3(2k + xk_x + yk_y) * P. \end{aligned} \tag{23}$$

The second term in equation (10) takes a similar form:

$$k_t * P = k_\sigma \dot{\sigma} * P \tag{24}$$

$$= -\frac{1}{\sigma}(2k + yk_y + xk_x)\dot{\sigma} * P, \tag{25}$$

and noting that $u_3 - \frac{\dot{\sigma}}{\sigma} = -\frac{\dot{Z}}{Z - \mu_f}$ completes the proof. □

We now make several claims that will be used to prove the theorem. We begin by noting that the operator $M[I]$ can be expressed as a convolution $m * I$ with some compactly-supported m. While the correct terms for M and m are an operator and a filter, respectively, we will refer to m as an "operation" rather than a "filter" to emphasize that it is a computational object and to distinguish it from the physical, light-blocking filter κ at the camera's aperture.

Claim 1. *The blur kernel k and post-processing operation m are related in the frequency domain by $\hat{k}(\hat{r}, \hat{\theta}) = f(\hat{\theta}) e^{-w \int_0^{\hat{r}} \frac{\hat{m}(s,\hat{\theta})}{s} ds}$ for $w(\sigma) = \frac{Z - \mu_f}{\dot{Z}} v$ and some angular function $f(\hat{\theta})$.*

Proof The Fourier transform takes the convolution

$$v\ m * k * P = R \tag{26}$$

to a multiplication

$$v\ \hat{m}\ \hat{k}\ \hat{P} = \hat{R}, \tag{27}$$

with hats indicating the Fourier transforms of the original distributions, expressed in polar coordinates $(\hat{r}, \hat{\theta}) = (\sqrt{\omega_x^2 + \omega_y^2}, \tan^{-1}(\omega_x, \omega_y))$. See appendix A for details on this use of the convolution theorem.

The Fourier transform of the residual takes the form

$$\hat{R} = \mathcal{F}\left[\frac{\dot{Z}}{Z - \mu_f} (2k + xk_x + yk_y) * P \right] \tag{28}$$

$$= \frac{\dot{Z}}{Z - \mu_f}\ \mathcal{F}[(2k + xk_x + yk_y)]\ \hat{P}, \tag{29}$$

where

$$\mathcal{F}[(2k + xk_x + yk_y)] = 2\hat{k} + i\partial_{\omega_x}(i\omega_x\hat{k}) + i\partial_{\omega_y}(i\omega_y\hat{k}) \tag{30}$$

$$= -\omega_x\hat{k}_{\omega_x} - \omega_y\hat{k}_{\omega_y} \tag{31}$$

$$= -\hat{r}\hat{k}_{\hat{r}} \tag{32}$$

so we can rewrite equation (27) as

$$v\ \hat{m}\ \hat{k}\ \hat{P} = -\frac{\dot{Z}}{Z - \mu_f}\ \hat{r}\ \hat{k}_{\hat{r}}\ \hat{P}. \tag{33}$$

We require this to hold for all underlying scene textures by dropping the $\hat{P}$ term from either side. This leaves a simple partial differential equation on $\hat{k}$. Compactness of m guarantees that $\hat{m}$ is smooth, and integrability of k, xk, and yk guarantee that $\hat{k}$, $\hat{k}_{\omega_x}$, and $\hat{k}_{\omega_y}$ are continuous, so we solve this equation using integrating factors. □

Claim 2. *The Fourier transform of the post-processing operation, $\hat{m}$, takes the form $g(\hat{\theta})\hat{r}^n$ for some $n \in \mathbb{C}$ and angular function $g(\hat{\theta})$.*

Proof Recall that the post-processing operation is required to be depth-blind, so $\hat{m}$ cannot be a function of the depth-scaling factor σ. However, we require that equation (14) holds for the entire family of possible blur kernels k, which are depth-scaled versions of the physical aperture filter κ according to equation (7). In the frequency domain this depth scaling takes the form

$$\hat{k}(\hat{r}, \hat{\theta}) = \hat{\kappa}(\sigma\hat{r}, \hat{\theta}). \tag{34}$$

This means that we can introduce the functions

$$\alpha(\sigma\hat{r}, \hat{\theta}) = \ln\left(\frac{\hat{\kappa}(\sigma\hat{r}, \hat{\theta})}{f(\hat{\theta})}\right), \tag{35}$$

$$\beta(\sigma) = -w(\sigma), \tag{36}$$

$$\gamma(\hat{r}, \hat{\theta}) = \int_0^{\hat{r}} \frac{\hat{m}(s, \hat{\theta})}{s} ds, \tag{37}$$

and rewrite a slightly rearranged form of claim 1 as

$$\alpha(\sigma\hat{r}, \hat{\theta}) = \beta(\sigma)\gamma(\hat{r}, \hat{\theta}). \tag{38}$$

Considering what happens when $\hat{r} = 1$, we see that

$$\alpha(\sigma, \hat{\theta}) = \beta(\sigma)\gamma(1, \hat{\theta}), \tag{39}$$

so α is separable in $\hat{\theta}$. This separability can be seen in the general-$\hat{r}$ case by replacing σ with $\sigma\hat{r}$ for an alternate expression for α:

$$\alpha(\sigma\hat{r}, \hat{\theta}) = \beta(\sigma\hat{r})\gamma(1, \hat{\theta}). \tag{40}$$

Taking the $\hat{r}$ derivative of equations (38) and (40) and noting that they must be equal, we have

$$\frac{d}{d\hat{r}}\big(\beta(\sigma)\gamma(\hat{r}, \hat{\theta})\big) = \beta(\sigma)\gamma^{(1,0)}(\hat{r}, \hat{\theta}) \tag{41}$$

$$= \frac{d}{d\hat{r}}\big(\beta(\sigma\hat{r})\gamma(1, \hat{\theta})\big) = \sigma\beta'(\sigma\hat{r})\gamma(1, \hat{\theta}), \tag{42}$$

so that again considering the $\hat{r} = 1$ case, we find that

$$\beta(\sigma) = \frac{\gamma(1, \hat{\theta})}{\gamma^{(1,0)}(1, \hat{\theta})}\sigma\beta'(\sigma). \tag{43}$$

This is a separable ordinary differential equation that has the solution

$$\beta(\sigma) = c\sigma^{\frac{\gamma^{(1,0)}(1,\hat{\theta})}{\gamma(1,\hat{\theta})}} \tag{44}$$

for some constant c. Because $\beta(\sigma)$ cannot change with $\hat{\theta}$, the exponent must also be a constant, which we call n. These forms of β and n allow equation (42) to be rewritten

$$\gamma^{(1,0)}(\hat{r}, \hat{\theta}) = \frac{\sigma\beta'(\sigma\hat{r})}{\beta(\sigma)}\gamma(1, \hat{\theta}) \tag{45}$$

$$= n\hat{r}^{n-1}\gamma(1, \hat{\theta}) \tag{46}$$

$$= \hat{r}^{n-1}\gamma^{(1,0)}(1, \hat{\theta}). \tag{47}$$

This derivative in γ simply removes the integral in equation (37), so that we can rewrite the equation above as

$$\frac{m(\hat{r}, \hat{\theta})}{\hat{r}} = \hat{r}^{n-1}\frac{m(1, \hat{\theta})}{1}. \tag{48}$$

Introducing $g(\hat{\theta}) = \hat{m}(1, \hat{\theta})$ completes the proof of the claim. □

Claim 3. *The operation exponent n must be a positive integer.*

Proof The proof of this claim relies on concepts from complex analysis that are introduced in appendix A.

According to Schwartz's Paley-Weiner theorem, the Fourier transform of a compactly supported distribution has continuous derivatives of all orders at every point. The origin is a location of particular interest, because almost all choices of g and n will lead to a discontinuity there.

First note that $g(\hat{\theta})$ cannot vanish everywhere. In this case, the aperture filter implied by claim 1 would be a Dirac delta pinhole. This corresponds to the no-residual optical flow case, which cannot reveal depth, and violates the integrability requirement on κ.

If $\Re(n)$ is negative, then $\hat{r}^n$ will go to complex infinity at the origin, and if n is not an integer or has an imaginary part, repeated application of the power

rule shows that some derivative of $\hat{m}$ will have an exponent n' with $\Re(n') < 0$ and the same discontinuity. Specifically, the j^{th}-order derivative in $\hat{r}$ of $\hat{m}$ is

$$\frac{d^j \hat{m}}{d\hat{r}^j} = \begin{cases} 0, & n \in \mathbb{Z}^+,\ j > n \text{ or } n = 0, \\ g(\hat{\theta}) \frac{n!}{(n-j)!} \hat{r}^{n-j}, & \text{else.} \end{cases} \tag{49}$$

As $\hat{r}$ approaches zero, this derivative approaches complex infinity for $\Re(n-j) < 0$ (unless n is a positive integer or zero), has an essential singularity when $n-j$ is imaginary, and otherwise goes to zero. So, there is a discontinuity in some order derivative in $\hat{r}$ at the origin unless n is a nonnegative integer. Strictly speaking, it is discontinuities in the derivatives in ω_x and ω_y that are forbidden, but these follow directly, e.g. by Faà di Bruno's formula. Thus, n must be a nonnegative integer.

When $n = 0$, $\hat{m} = g(\hat{\theta})$ must take a constant value g to avoid a discontinuity at the origin. Note that $\hat{k}$ under unit depth scaling ($\sigma = 1$) is exactly $\hat{\kappa}$, and consider the corresponding filter implied by claim 1:

$$\hat{\kappa} = f(\hat{\theta}) e^{-w(1) g \ln(\hat{r})} = f(\hat{\theta}) \hat{r}^{-w(1)g}. \tag{50}$$

Because κ is integrable, the Riemann-Lebesgue lemma states that $\hat{\kappa}$ must vanish as $\hat{r}$ approaches infinity, so $\Re(w(1)g)$ must be positive. However, this implies an infinite discontinuity at the origin which also violates integrability assumptions: integrability of κ implies uniform continuity of $\hat{\kappa}$. Thus, $n \neq 0$. □

Claim 4. *$\hat{m}$ is a homogeneous polynomial of degree n.*

Proof Combining the previous claims, we have that for some positive integer n,

$$\hat{\kappa}(\hat{r}, \hat{\theta}) = f(\hat{\theta})\ e^{-w(1) g(\hat{\theta}) \hat{r}^n}. \tag{51}$$

By the Riemann-Lebesgue lemma, integrability of κ implies $\hat{\kappa}$ vanishes at infinity, which requires $\Re(w(1)g(\hat{\theta})) > 0$ for $\hat{\theta} \in [-\pi, \pi]$. Then, $g(\hat{\theta}) \neq 0$ on $[-\pi, \pi]$ and in both ω_x and ω_y, $\hat{m}$ has a pole at (complex) infinity.

Schwartz's Paley-Wiener theorem states that the Fourier transform of a compactly supported distribution can be extended to an entire function, i.e. one that is complex differentiable everywhere in $\mathbb{C}^2$. Proofs of the previous claims have used the smoothness of $\hat{m}$ that this theorem implies over real values of ω_x and ω_y, but this is a much more restrictive condition; the function $\Re(z)$, for example, is nowhere complex differentiable. In fact, the only entire functions with a pole at infinity are polynomials. See appendix A for details. Because $\hat{m}$ is degree n along any radial slice, it must also be homogeneous. □

Claim 5. *$f(\hat{\theta}) = a_0 \in \mathbb{R}^+$ and $n = 2$.*

Proof Combining the previous claims, we have that for some positive integer n, and constants $c_0, ..., c_n \in \mathbb{R}$

$$\hat{\kappa}(\hat{r}, \hat{\theta}) = f(\hat{\theta})\ e^{-w(1) \sum_j c_j \omega_x^j \omega_y^{n-j}}. \tag{52}$$

Integrability of κ implies that $\hat{\kappa}$ is uniformly continuous, so $f(\hat{\theta})$ must be a constant a_0 to avoid a discontinuity in $\hat{\kappa}$ at the origin, where the exponential term goes to one. It must be real by the conjugate symmetry of $\hat{\kappa}$ induced by reality of κ, and it cannot be negative or zero without causing κ to be so as well.

The Riemann-Lebesgue lemma implies that n is even, because $\hat{\kappa}$ must vanish as each ω approaches either positive or negative infinity while the sign of each c_j is fixed. For n even,

$$\begin{aligned} g(\hat{\theta}) &= \sum_j c_j \cos^j(\hat{\theta}) \sin^{n-j}(\hat{\theta}) \\ &= \sum_j c_j (-1)^j \cos^j(\hat{\theta} + \pi)(-1)^{n-j} \sin^{n-j}(\hat{\theta} + \pi) \\ &= g(\hat{\theta} + \pi) \end{aligned} \tag{53}$$

Next see that for $n \geq 3$, $\hat{\kappa}$ along any fixed-$\hat{\theta}$ radial slice is not a positive definite function, because with

$$C(n) = \sum \sum z_i z_j a_0 e^{-w(1) g(\hat{\theta}) |r_i - r_j|^n}, \tag{54}$$

$$z = [1,\ -2,\ 1], \tag{55}$$

$$r = \left[-\sqrt[n]{\frac{.1}{w(1)g(\hat{\theta})}},\ 0,\ \sqrt[n]{\frac{.1}{w(1)g(\hat{\theta})}} \right], \tag{56}$$

we have

$$C(n) = 6 - 8e^{-.1} + 2e^{-(.1)(2^n)}, \tag{57}$$

and both $C(3)$ and $\frac{dC}{dn}$ are negative. The Fourier slice theorem [2,21] states that each of these angular slices is the one-dimensional Fourier transform of the projection of κ along the same angle in the spatial domain:

$$\hat{\kappa}(\hat{r}, \theta_0) = \mathcal{F}_{1D} \left[\int \kappa(r \cos\theta_0 + z \sin\theta_0, r \sin\theta_0 - z \cos\theta_0) dz \right]. \tag{58}$$

However, Bochner's theorem states that the Fourier transform of a nonnegative integrable function must be positive definite. So for $n \geq 3$, all projections of κ, and therefore the filters κ themselves, fail to meet our requirement of nonnegativity and integrability. □

Proof (Theorem) Combining the previous claims we see that

$$\hat{m} = c_2\omega_x^2 + c_1\omega_x\omega_y + c_0\omega_y^2 \tag{59}$$

$$\hat{\kappa} = a_0 e^{-w(1)(c_2\omega_x^2 + c_1\omega_x\omega_y + c_0\omega_y^2)}. \tag{60}$$

Integrability of κ requires that $w(1) > 0$ and $c_1^2 < 4c_0c_2$, so that for

$$\Sigma^2 = w(1), \tag{61}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} c_0 & -c_1/2 \\ -c_1/2 & c_2 \end{bmatrix}, \tag{62}$$

$$a = \frac{a_0}{2\Sigma\sqrt{|\boldsymbol{\Sigma}|}}, \tag{63}$$

the inverse Fourier transforms and basic manipulation prove the theorem. □

3 Inherent Sensitivity

Due to the loss of image contrast as an object moves away from the focal plane, we expect the focal flow depth signal to be strongest for scene patches that are in focus or nearly in focus. This is similar to the expected performance of stereo or depth from defocus, for which depth accuracy degrades at large distances. In those cases, accuracy is enhanced by increasing the baseline or aperture size. In focal flow, focal settings play the analogous role.

Following Schechner and Kiryati in [28], we can describe the inherent sensitivity of all three depth cues. Recall that for a stereo system with baseline b and an inference algorithm that estimates disparity Δx, depth is measured as

$$Z = \frac{b\mu_s}{\Delta x}, \tag{64}$$

with first-order sensitivity to the disparity estimate

$$\left|\frac{dZ}{d(\Delta x)}\right| = \left|\frac{b\mu_s}{-(\Delta x)^2}\right| = \frac{Z^2}{b\mu_s}. \tag{65}$$

Similarly, for a depth from defocus sensor with aperture radius A and an algorithm that estimates blur radius $\tilde{A}$, the sensitivity of depth to error in $\tilde{A}$ is

$$Z = \frac{\mu_f\mu_s A}{\mu_f\tilde{A} + \mu_s A}, \tag{66}$$

$$\left|\frac{dZ}{d\tilde{A}}\right| = \left|\frac{-\mu_f^2\mu_s A}{(\mu_f\tilde{A} + \mu_s A)^2}\right| = \frac{Z^2}{A\mu_s}. \tag{67}$$

These equations show a fundamental similarity between stereo and depth from defocus, in which the baseline and aperture size are analogous.

For a toy model of focal flow, we consider images of a sinusoidal texture blurred by a normalized and radially-symmetric Gaussian. Note that for a circular lens, a radially-symmetric filter will provide the best performance because it will have the highest light efficiency. We assume the texture has frequency ω_0, unit amplitude, and arbitrary phase and orientation. The image captured at time t has frequency ω and amplitude B, which are determined by depth (see appendix B.2 for additional detail):

$$\omega(t) = Z\omega_0/\mu_s, \tag{68}$$

$$B(t) = \iint \frac{e^{-\frac{x^2}{4\Sigma^2\sigma^2}}}{4\pi\Sigma^2\sigma^2}\cos(\omega(t)x)dxdy = e^{-\Sigma^2\omega_0^2\left(\frac{Z-\mu_f}{\mu_f}\right)^2}. \tag{69}$$

Depth can be measured from image amplitude, frequency, and their derivatives:

$$Z = \frac{\mu_f}{1 + \left(\frac{\mu_f}{\mu_s\Sigma}\right)^2 \frac{\dot{B}}{2B\omega\dot{\omega}}}. \tag{70}$$

When image quantities $(\omega, \dot{\omega}, B, \dot{B})$ are measured within error bounds $(\epsilon_\omega, \epsilon_{\dot{\omega}}, \epsilon_B, \epsilon_{\dot{B}})$, a simple propagation of uncertainty bounds the depth error ϵ_Z:

$$\epsilon_Z \leq \sqrt{\left(\frac{\partial Z}{\partial\omega}\right)^2\epsilon_\omega^2 + \left(\frac{\partial Z}{\partial\dot{\omega}}\right)^2\epsilon_{\dot{\omega}}^2 + \left(\frac{\partial Z}{\partial B}\right)^2\epsilon_B + \left(\frac{\partial Z}{\partial B_1}\right)^2\epsilon_{\dot{B}}^2} \tag{71}$$

$$= \frac{Z|Z-\mu_f|}{\mu_f}\sqrt{\frac{\epsilon_\omega^2}{\omega^2} + \frac{\epsilon_{\dot{\omega}}^2}{\dot{\omega}^2} + \frac{\epsilon_B^2}{B^2} + \frac{\epsilon_{\dot{B}}^2}{\dot{B}^2}}. \tag{72}$$

The sum of error terms in the radicand describes the relative usefulness of improving accuracy in either brightness or spatial frequency measurements for a given scene. It could guide the design or selection of an optimized photosensor, e.g. [38], because when combined with an appropriate statistical model of the scenes to be imaged, it quantifies the trade-off between bit depth, which places a lower bound on ϵ_B and $\epsilon_{\dot{B}}$, and spatial resolution, which likewise bounds ϵ_ω and $\epsilon_{\dot{\omega}}$.

Depending on the error model, the radicand in expression (72) could introduce additional scene dependencies, but in the simplest case, it is constant and focal flow is immediately comparable to stereo and depth from defocus. Just as the sensitivity of those measurements goes as depth squared, we see that focal flow measurements are sensitive to object distance from both the camera and the focal plane through the $Z|Z-\mu_f|$ term. The focal flow analogue to aperture size or baseline in this scenario is inverse magnification, the ratio of in-focus depth μ_f to sensor distance μ_s.

4 Prototype and Evaluation of Non-idealities

In theory, when an ideal thin lens camera with an infinitely-wide Gaussian aperture filter observes a single moving, front-parallel, textured plane, there is a unique solution $\boldsymbol{v} \in \mathbb{R}^4$ to the system of per-pixel linear focal flow constraints (equation (17)), and this uniquely resolves the scene depth $Z(\boldsymbol{v})$ and velocity $(\dot{X}, \dot{Y}, \dot{Z})(\boldsymbol{v})$ through equations (18) and (19). In practice, a physical instantiation of a focal flow sensor will deviate from the idealized model, and there will only be approximate solutions $\tilde{\boldsymbol{v}} \in \mathbb{R}^4$ that can produce errors in depth and velocity measurements.

We expect two main deviations from the idealized model. First, thick lenses have optical aberrations and a finite extent, making it impossible to create ideal Gaussian blur kernels that scale exactly with depth. Second, image derivatives must be approximated by finite differences between noisy photosensor values. We assess the impacts of both of these effects using the prototype shown in appendix C. Based on 1"-diameter radially symmetric optics, it includes an f=100mm planar-convex lens, a monochromatic camera (Grasshopper GS3-U3-23S6M-C, Point Grey Research), and an adjustable-length lens tube. The aperture side of the sensor supports various configurations, including an adjustable aperture diaphragm and the optional inclusion of a Gaussian apodizing filter (NDYR20B, Thorlabs) adjacent to the planar face of the lens. A complete list of parts can be found in figure 6 in appendix C. Because the optics are radially symmetric, $\boldsymbol{\Sigma}$ is assumed proportional to the identity matrix and the blur kernels can be parameterized by the scalar Σ.

To accommodate these non-idealities, we formulate the patch-based computation of depth and velocity as a feed-forward sequence of computations that operates on all pixels in parallel. Each step in the computational sequence is differentiable, so all of the tunable parameters can be optimized by capturing images of textured targets whose depth is known, and then back-propagating the depth errors to update the parameters by gradient descent. We use this to calibrate the values of the two optical parameters in Section 4.2, and then we extend it to simultaneously optimize the coefficients of the discrete derivative operators in Section 4.5.

4.1 Feed-forward, patch-wise computation

For all results, we produce depth and velocity measurements using the simple sequence of computations that follows directly from the modeling of Section 2. The input is three ordered frames from a temporal sequence, $I(x, y, t_i), i \in \{1, 2, 3\}$, and the computation has four steps: (i) approximate spatial and temporal derivatives using discrete kernels, $I_x = D_x * I$, $I_y = D_y * I$, $I_t = D_t * I$, and $(I_{xx} + I_{yy}) = D_{\nabla^2} * I$; (ii) aggregate the per-pixel linear constraints (Equation (14)) using sliding window to create per-pixel matrix equations $A\mathbf{v} = \mathbf{b}$; (iii) assemble and invert the corresponding normal equations $A^T A\mathbf{v} = A^T\mathbf{b}$ at all pixels; and (iv) compute per-pixel depth and 3D velocity using Equations (18, 19).

A reference Matlab implementation that computes depth and velocity at all pixels in parallel is available on the project website.[1] Note that the measurement process requires knowing the image sensor's principal point (the origin of the coordinate system for x and y in equation (17)), and we align this to the center of the sensor during assembly. The remaining tunable parameters of the computational sequence include the coefficients of the derivative filters $\{D_x, D_y, D_t, D_{\nabla^2}\}$, the spatial windowing filter, and the two optical parameters shown in Figure 2: sensor distance μ_s and aperture width Σ. (The object focal distance μ_f is determined by the lens' known focal length, $f = 100$mm.)

4.2 Depth-based calibration of optical parameters

We begin by setting the derivative and windowing kernels manually and optimizing the optical parameters. We use temporal kernel $I_t(x, y) \approx 1/2\,(I(x, y, t_3) - I(x, y, t_1))$, and spatial kernels $D_x = (-1/2, 0, 1/2)$, $D_{xx} = D_x * D_x$ (likewise in y) convolved with the middle frame $I(x, y, t_2)$. These spatial derivatives are sensitive to noise, so we emulate a lower-noise sensor by creating each input frame as the average of ten shots from the camera unless otherwise noted. We also find that numerical stability is improved by pre-normalizing the spatial coordinates $x \leftarrow x/c, y \leftarrow y/c$ for some constant c (we use $c = 10^4$). This pre-normalization and the use of finite differences lead to depth and velocity values that, if computed naively with equations (18) and (19), are scaled by an unknown constant, but this is naturally accounted for by the depth-based calibration we describe here.

To acquire calibration data, we mount a textured plane on a high-precision translation stage and laser align it to be normal to the sensor's optical axis. We collect images of this texture at known locations Z^i and with unknown sensor locations μ_s^j. We densely sample texture locations Z^i, so that the most in-focus image reveals lens tube length $\Delta\mu_s^j$, determining sensor locations μ_s^j up to a shared constant μ_s^0.

[1] http://vision.seas.harvard.edu/focalflow/

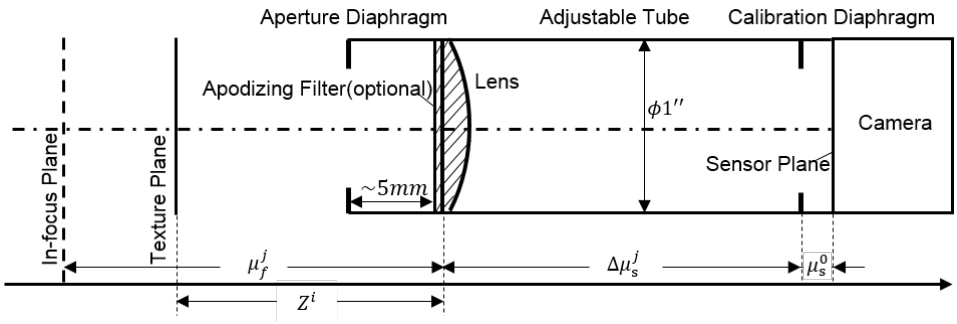


Fig. 2 Calibration Parameters. Depth planes and parameters used to calibrate prototype sensor, see section 4.2.

We run the feed-forward computation on every triple of frames in the calibration dataset, producing a single scalar estimate Z_{est} from each triple by aggregating constraints over a single large (201×201) window in the center of the frame. We optimize the optical parameters Σ and μ_s by gradient descent using a loss based on the RMS depth error,

$$\Sigma, \mu_s^0 = \arg\min_{\Sigma,\mu_s^0} \sum_{i,j} \rho(Z_{est}(I^{ij}; \Sigma, \mu_s^0 + \Delta\mu_s^j) - Z^i), \tag{73}$$

where $\rho(x) = \{x^2 \text{ if } |x| \leq 1, \text{and } 1 \text{ otherwise}\}$ is a robust functional that reduces the effect of outliers. Figure 3 shows a typical cost surface for this objective. Experimentally, we observe convergence to a good extremum for a wide range of initializations.

Note that this calibration must be repeated when the aperture is reconfigured, such as when inserting an apodizing filter or adjusting the diaphragm. When the effective blur kernels change, so does the optimal effective width Σ. But for a fixed aperture, we find that the sensor distance μ_s can be adjusted without re-calibrating Σ.

While developing this calibration procedure, we tried alternatives that led to inferior results. In particular, we tried calibrating the sensor distance μ_s using conventional methods, and then measuring the point spread functions $k(x, y, Z_i)$ for depths Z_i (e.g., right of Figure 4) and fitting parameter Σ according to the Gaussian thin lens model: $\kappa(r) = \frac{e^{-r^2/4\Sigma^2\sigma(Z_i)^2}}{\sigma(Z_i)^2}$. This approach is less effective, especially when the point spread functions deviate substantially from Gaussians, because it optimizes a fit to the points spread functions instead of depth accuracy, which is what we care about most. Also, unlike optimizing with respect to depth, it does not allow for the simultaneous optimization of both optical parameters and derivative kernels, which is explored in Section 4.5.

4.3 Results

Figures 4 and 5 show performance for different apertures and noise levels. Accuracy is determined using a textured front-parallel plane whose ground truth position and velocity are precisely controlled by a translation stage. In each case, the measurement algorithm is applied to a 201×201 window around the center of the 960×600 image. The top and middle rows of Figure 4 compare the measured depth Z and speed $\|(\dot{X}, \dot{Y}, \dot{Z})\|$ to ground truth, indicated by solid black lines. Speed is measured in units of millimeter per video frame (mm/frame). Different colors in these plots represent experiments with different in-focus distances μ_f, corresponding to different lengths of the adjustable lens tube. We show measurements taken both with an apodizing filter (and open diaphragm) and without it (with diaphragm closed to about $\varnothing 4.5$mm). In both cases, the inset point spread functions reveal a deviation from the Gaussian ideal, but the approximate solutions to the linear constraint equations still provide useful depth in-

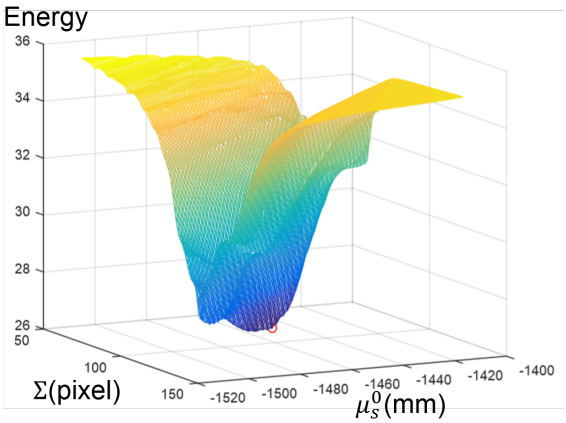


Fig. 3 A typical calibration cost surface. The surface of energy function (equation (73)) has an optimal value (red circle) over a wide range of feasible region, which ensures the convergence of the optimization in the calibration process.

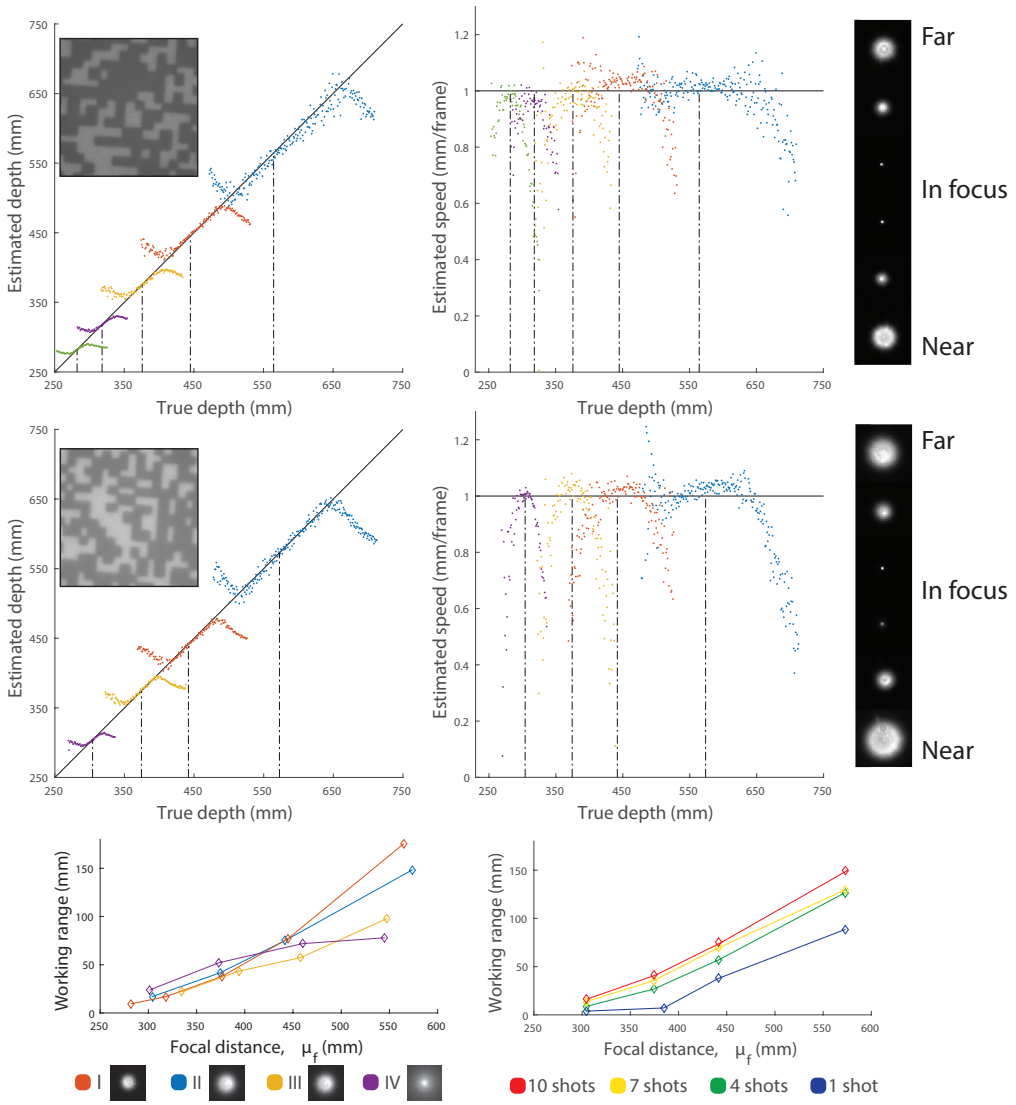


Fig. 4 Accuracy and working range. *Top and middle rows*: Estimated depth and speed versus true depth for two aperture settings: open ⌀4.5 diaphragm (*top*) and apodizing filter (*middle*). Solid black lines are true depth and speed. Insets are sample image and PSF. Colors are separate trials with different focal distances μ_f, marked by dashed vertical lines. Depth interval for which depth error is less than 1% of μ_f defines the working range. *Bottom left*: Sample PSFs, and working range versus focal distance, for aperture settings: (I) diaphragm ⌀4.5mm, no filter; (II) diaphragm open, with filter; (III) diaphragm ⌀8.5mm, no filter; (IV) diaphragm ⌀25.4mm, no filter. We observe that larger apertures correspond to smaller working ranges. *Bottom right*: Working range for distinct noise levels, controlled by number of averaged shots.

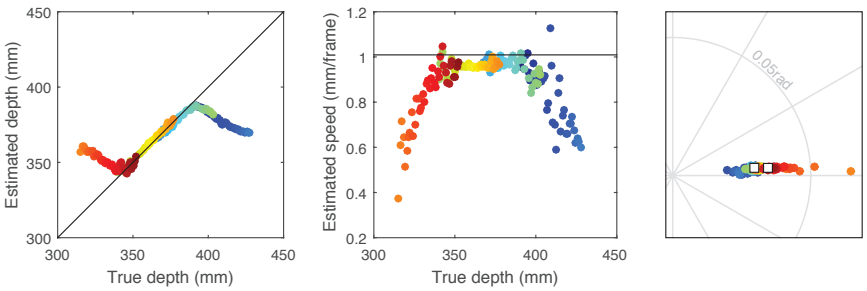


Fig. 5 Velocity. Measured depth, speed, and 3D direction $(\dot{X}, \dot{Y}, \dot{Z})/\|(\dot{X}, \dot{Y}, \dot{Z})\|$ versus true depth, with markers colored by true depth. Directions shown by orthographic projection to XY-plane, where the view direction is the origin. Ground truth is black lines for depth and speed, and white squares for direction. (Two ground truth directions result from remounting a translation stage to gain sufficient travel.)

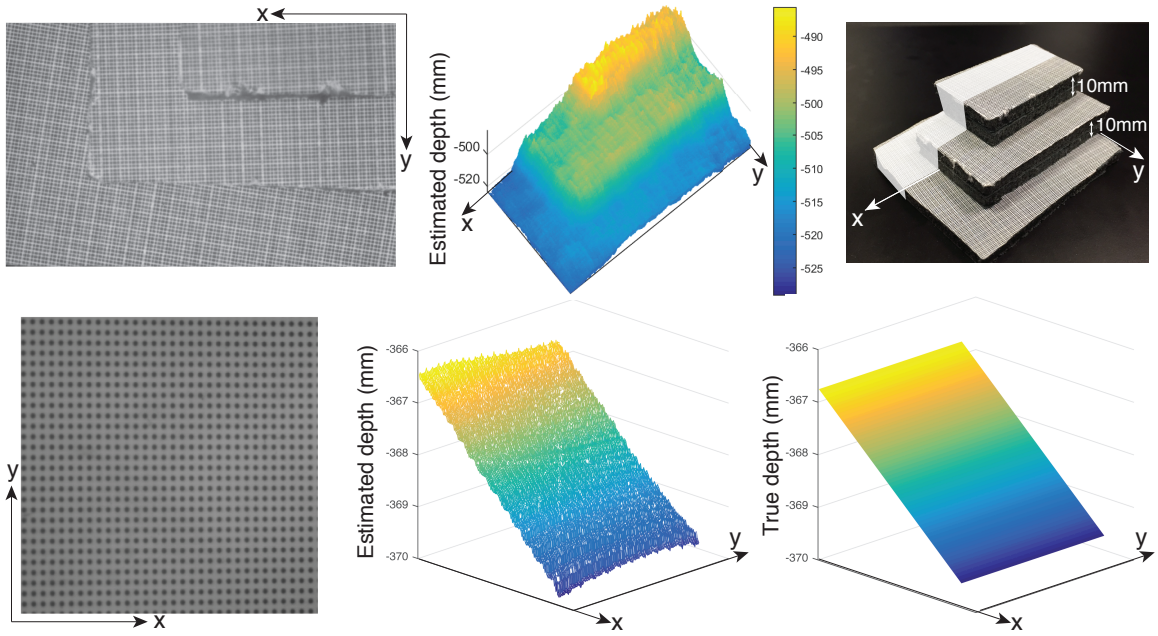


Fig. 6 Depth maps for two different scenes. From left to right: one frame from an input three-frame image sequence; per-pixel depth measured by independent focal flow reconstruction in overlapping square windows; and true scene shape.

formation over ranges that are roughly centered at, and proportional to, the focal distances.

The bottom of Figure 4 shows the effects that aperture configuration and noise level have on the working range, defined as the range of depths for which the absolute difference between the measured depth and the true depth is less than 1% of the focal distance μ_f. The prototype achieves a working range of more than 15cm. Figure 5 shows both the measured speed and the measured 3D direction of a moving texture. Comprehensive results for different textures, aperture configurations, and noise levels can be found in appendix D.

Figure 6 shows full-field depths maps measured by the system. Each is obtained by applying the reconstruction algorithm in parallel to overlapping windows. We used 71×71 windows for the top row and 241×241 windows for the bottom, again on 960×600 images. We do not use multiple window sizes or any form of spatial regularization; we simply apply the reconstruction algorithm to every window independently. Even using this simple approach, the depths map are consistent with the scene's true shape, even when the shape is not front-parallel. The Matlab code used to generate these depth maps can be found in our project website. It executes in 6.5 seconds on a 2.93GHz processor with Intel Xeon $X5570$ CPU.

4.4 Empirical comparison with single-shot DFD

Another way to extract depth with an unactuated, monocular sensor is single-shot depth from defocus with a binary coded aperture (e.g., [33,18,36]), where one explicitly deconvolves each image patch with a discrete set of per-depth blur kernels and selects the most "natural" result. Compared to focal flow, this provides a larger working range, but lower depth precision and a much greater computational burden.

We compare the performance of focal flow and Levin et al.'s [18] single-shot depth from defocus in simulation. The simulation used sensor dimensions $f = 100$mm and $\mu_s = 130$mm for both approaches. For single-shot depth-from-defocus, we used the binary aperture pattern from [18]. For focal flow, we used radially-symmetric Gaussian blur kernels, and all other settings were the same as those used in section 4.3. Zero-mean Gaussian noise with variance 10^{-6} was added to the simulated input images for both methods. We used a randomly selected texture from the CuRET database[2][4], and to capture the best possible performance of the single-shot approach, we used the same texture for the training step (parameters λ_k) of [18]. For focal flow, we simulated input images for depths between 400mm and 500mm at increments of 1mm; and for the single-shot approach, we simulated input images for depths between 320mm and 720mm with increments of 4mm. As in section 4.3, the working range (400mm-500mm) of focal flow is de-

[2] http://www.cs.columbia.edu/CAVE/software/curet/

termined as the set of depths for which the absolute depth error is less than 1% of the focal distance μ_f. For each approach, we obtained depth estimates for each of the 101 increments, and the RMS depth error was computed over these estimates.

The depth performance of the two approaches is shown in Table 1. It includes evaluations for each of the three deconvolution algorithms proposed in [18], as implemented in Matlab by the authors.[3] The depth performance of focal flow and single-shot depth from defocus is complimentary: the working volume of the single-shot approach is four times larger, but focal flow is more than seven times as precise and at least hundreds of times faster.

4.5 End-to-end Optimization

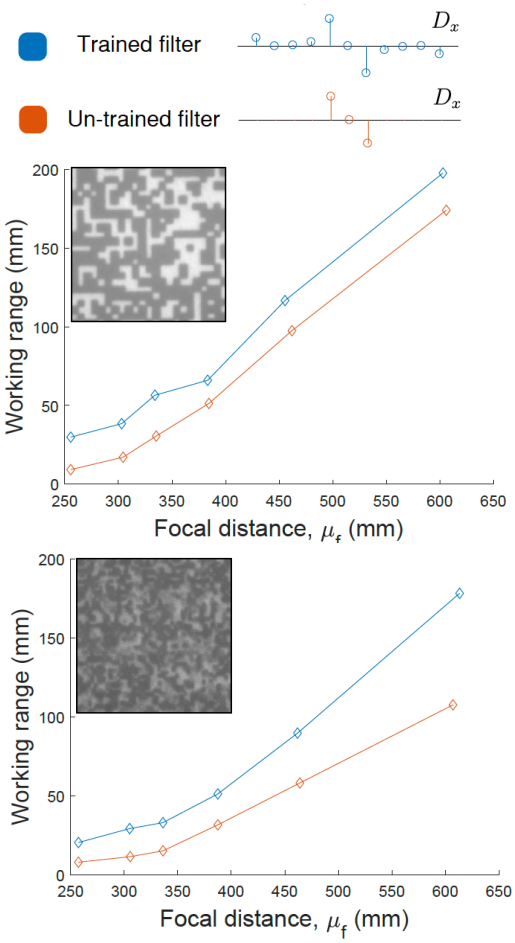


Fig. 7 Working range improved with end-to-end training. Using trained derivative filters (blue) instead of finite differences (red) extends the working range($\equiv 1\%\mu_f$) by at least 10mm. Results on two different test textures are shown.

[3] https://groups.csail.mit.edu/graphics/CodedAperture/

Next, we extend the calibration of optical parameters Σ and μ_s by also optimizing the finite difference kernels D_x, D_y, D_{xx}, D_{yy}. This allows learning kernels that incorporate low-pass filtering to suppress sensor noise, while also adjusting the optical parameters to account for the implicit changes in spatial scale that these kernels induce. It is possible because each step in our computational sequence is differentiable, so the derivatives of our depth-based loss can be back-propagated to simultaneously adjust all of the optical and computational parameters.

In this experiment, we first optimize the optical parameters μ_s and Σ with naive derivative kernels as described in Section 4.2, and then we fine tune Σ and the spatial derivative kernels D_x, D_y, D_{xx}, D_{yy} to minimize the 1-norm loss:

$$L(Z_{est} - Z_{true}) = \sum |Z_{est} - Z_{true}|. \tag{74}$$

We initialize $D_x = [0, 0, 0, 0, -0.5, 0, 0.5, 0, 0, 0, 0]$ and enforce the constraint that D_x (and hence $\frac{\partial L}{\partial D_x}$) is anti-symmetric during optimization. We also force $D_y = D_x^T$, $D_{xx} = D_x * D_x$, so the total number of free parameters in the derivative kernels is 6. For faster convergence, we multiply a manually-set constant λ_Σ to $\frac{\partial L}{\partial \Sigma}$, to keep $\left(\lambda_\Sigma \frac{\partial L}{\partial \Sigma}\right)/\Sigma$ to be smaller than, or roughly the same order as, $\left(\frac{\partial L}{\partial D_x}\right)/D_x$. We used $\lambda_\Sigma = 10^{-6}$ in our experiment. We use the same 10-shot averaging calibration data as in Section 4.2.

Figure 7 shows the effect of the optimized kernels on two of the test textures. The working range can be extended by at least 10mm for each μ_f and each texture by training the filters. Although the noise is already partially suppressed in the 10-shot data, we still see a gain in working range. We predict a larger performance boost from this training procedure on noisier images.

5 Discussion

By combining blur and differential motion in a way that mitigates their individual weaknesses, focal flow enables a passive, monocular sensor that provides depth and 3D velocity from a simple, small-patch measurement algorithm. While the focal flow theory is developed using Gaussian blur kernels and front-parallel scene patches, we find in practice that it can provide useful scene information for a much broader class of aperture configurations, and some slanted scene planes.

The prototype described in this paper currently has some limitations. Its simple measurement algorithm performs independent measurement in every local patch. As such, it is overly sensitive to noise and requires high-contrast texture to be everywhere in the scene.

Table 1 Focal flow vs. single-shot depth from defocus. We compare speed, accuracy, and working range of focal flow and Anat et al. [18] in simulation. Focal flow requires multiple images and has a smaller working range but runs more than 100x faster with less than 1/7 error.

Method	Running Time (sec/estimate)	Working Range (mm)	RMS depth error (mm)
Focal Flow	0.03	400-500	2.94
Coded DfD: L2 deconvolution in frequency domain, we = 0.01	7.90	320-720	20.95
Coded DfD: L2 deconvolution in frequency domain, we = 0.002	7.93	320-720	44.73
Coded DfD: L2 deconvolution we = 0.01, max_it = 80	159.03	320-720	56.25
Coded DfD: sparse deconvolution we = 0.01, max_it = 200	1456.62	320-720	45.24

Performance can likely be improved by including noise suppression and dynamical filtering that combines the available depth and velocity values. At the expense of additional computation, performance could be improved by adapting techniques from optical flow and stereo, such as outlier-rejection, multi-scale reasoning, and spatial regularization that can interpolate depth in textureless regions. Projecting a pattern onto the scene could also provide texture at the expense of electrical power.

The relative efficiency of focal flow suggests its suitability for small, low-power platforms, particularly those with well-defined working ranges and regular ambient motion, either from the platform or the scene. While the prototype shown here is relatively large and slow, an optimized processor and miniaturized optics could greatly benefit microrobots for which traditional computer vision has had little to offer.

6 Acknowledgments

We would like to thank J Zachary Gaslowitz and Ioannis Gkioulekas for helpful discussion. This work was supported by a gift from Texas Instruments Inc. and by the National Science Foundation under awards No. IIS-1212928 and 1514154 and Graduate Research Fellowship No. DGE1144152 to E.A.

References

1. Alexander, E., Guo, Q., Koppal, S., Gortler, S., Zickler, T.: Focal flow: Measuring distance and velocity with defocus and differential motion. In: European Conference on Computer Vision, pp. 667–682. Springer (2016)
2. Bracewell, R.N.: Strip integration in radio astronomy. Australian Journal of Physics **9**(2), 198–217 (1956)
3. Chakrabarti, A., Zickler, T.: Depth and deblurring from a spectrally-varying depth-of-field. In: European Conference on Computer Vision (ECCV) (2012)
4. Dana, K.J., van Ginneken, B., Nayar, S.K., Koenderink, J.J.: In: Reflectance and Texture of Real-world Surfaces
5. Duhamel, P.E.J., Perez-Arancibia, C.O., Barrows, G.L., Wood, R.J.: Biologically inspired optical-flow sensing for altitude control of flapping-wing microrobots. IEEE/ASME Transactions on Mechatronics **18**(2), 556–568 (2013)
6. Farid, H., Simoncelli, E.P.: Range estimation by optical differentiation. Journal of the Optical Society of America A **15**(7), 1777–1786 (1998)
7. Favaro, P., Burger, M., Soatto, S.: Scene and motion reconstruction from defocused and motion-blurred images via anisotropic diffusion. In: European Conference on Computer Vision (ECCV) (2004)
8. Fisher, S.D.: Complex variables. Courier Corporation (1999)
9. Floreano, D., Zufferey, J.C., Srinivasan, M.V., Ellington, C.: Flying insects and robots. Springer (2009)
10. Grossmann, P.: Depth from focus. Pattern Recognition Letters **5**(1), 63–69 (1987)
11. Horn, B.K., Fang, Y., Masaki, I.: Time to contact relative to a planar surface. In: Intelligent Vehicles Symposium (IV) (2007)
12. Horn, B.K., Fang, Y., Masaki, I.: Hierarchical framework for direct gradient-based time-to-contact estimation. In: Intelligent Vehicles Symposium (IV) (2009)
13. Horn, B.K., Schunck, B.G.: Determining optical flow. In: 1981 Technical Symposium East. International Society for Optics and Photonics (1981)
14. Humber, J.S., Hyslop, A., Chinn, M.: Experimental validation of wide-field integration methods for autonomous navigation. In: Intelligent Robots and Systems (IROS) (2007)
15. Koppal, S.J., Gkioulekas, I., Zickler, T., Barrows, G.L.: Wide-angle micro sensors for vision on a tight budget. In: Computer Vision and Pattern Recognition (CVPR) (2011)
16. Lee, D.N.: A theory of visual control of braking based on information about time-to-collision. Perception (5), 437–59 (1976)
17. Levin, A.: Analyzing depth from coded aperture sets. In: European Conference on Computer Vision (ECCV) (2010)
18. Levin, A., Fergus, R., Durand, F., Freeman, W.T.: Image and depth from a conventional camera with a coded aperture. In: ACM Transactions on Graphics (TOG) (2007)
19. Lin, H.Y., Chang, C.H.: Depth from motion and defocus blur. Optical engineering **45**(12), 127,201–127,201 (2006)
20. Myles, Z., da Vitoria Lobo, N.: Recovering affine motion and defocus blur simultaneously. Pattern Analysis and Machine Intelligence (6), 652–658 (1998)

21. Ng, R.: Fourier slice photography. In: ACM Transactions on Graphics (TOG) (2005)
22. Paramanand, C., Rajagopalan, A.N.: Depth from motion and optical blur with an unscented kalman filter. IEEE Transactions on Image Processing **21**(5), 2798–2811 (2012)
23. Pentland, A.P.: A new sense for depth of field. Pattern Analysis and Machine Intelligence (4), 523–531 (1987)
24. Raghavendra, C.S., Sivalingam, K.M., Znati, T.: Wireless sensor networks. Springer (2006)
25. Rajagopalan, A., Chaudhuri, S.: Optimal selection of camera parameters for recovery of depth from defocused images. In: Computer Vision and Pattern Recognition (CVPR) (1997)
26. Rajagopalan, A., Chaudhuri, S., Mudenagudi, U.: Depth estimation and image restoration using defocused stereo pairs. Pattern Analysis and Machine Intelligence **26**(11), 1521–1525 (2004)
27. Rudin, W.: Functional analysis. McGraw-Hill (1991)
28. Schechner, Y.Y., Kiryati, N.: Depth from defocus vs. stereo: How different really are they? International Journal of Computer Vision **39**(2), 141–162 (2000)
29. Seitz, S.M., Baker, S.: Filter flow. In: International Conference on Computer Vision (ICCV) (2009)
30. Sellent, A., Favaro, P.: Coded aperture flow. In: German Conference on Pattern Recognition (GCPR) (2014)
31. Subbarao, M.: Parallel depth recovery by changing camera parameters. In: International Conference on Computer Vision (ICCV) (1988)
32. Subbarao, M., Surya, G.: Depth from defocus: A spatial domain approach. International Journal of Computer Vision **13**(3), 271–294 (1994)
33. Tai, Y.W., Brown, M.S.: Single image defocus map estimation using local contrast prior. In: International Conference on Image Processing (ICIP) (2009)
34. Tang, H., Cohen, S., Price, B., Schiller, S., Kutulakos, K.N.: Depth from defocus in the wild. In: Computer Vision and Pattern Recognition (CVPR) (2017)
35. Tao, M., Hadap, S., Malik, J., Ramamoorthi, R.: Depth from combining defocus and correspondence using light-field cameras. In: International Conference on Computer Vision (ICCV) (2013)
36. Veeraraghavan, A., Raskar, R., Agrawal, A., Mohan, A., Tumblin, J.: Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. In: ACM Transactions on Graphics (TOG) (2007)
37. Watanabe, M., Nayar, S.K.: Rational filters for passive depth from defocus. International Journal of Computer Vision **27**(3), 203–225 (1998)
38. Yasuma, F., Mitsunaga, T., Iso, D., Nayar, S.K.: Generalized assorted pixel camera: postcapture control of resolution, dynamic range, and spectrum. IEEE Transactions on Image Processing **19**(9), 2241–2253 (2010)
39. Zhou, C., Lin, S., Nayar, S.: Coded aperture pairs for depth from defocus. In: International Conference on Computer Vision (ICCV) (2009)
40. Zhou, C., Lin, S., Nayar, S.K.: Coded aperture pairs for depth from defocus and defocus deblurring. International Journal of Computer Vision **93**(1), 53–72 (2011)

Appendix A: Introduction to Distributions and Complex Analysis

The theory of distributions generalizes functional analysis beyond the classic notion of a function. Much of its usefulness comes from extending results from the study of differential equations to include discontinuous objects whose derivatives do not exist in the traditional sense, but can nonetheless be abstracted in consistent and powerful ways. While standard functions and measures can be treated as distributions, so can more exotic objects like the Dirac delta "function", which is the distributional derivative of the discontinuous step function, as well as its derivatives in turn.

In this context, a distribution is defined as a linear functional that maps some set of well-behaved functions to the real numbers. Unlike functions, they do not have values at given points in a domain, though this can be a useful way to visualize their effect. Any locally-integrable function P can induce a distribution $\tilde{P}$ that maps a good test function f (more detail below) to the reals through integration:

$$\langle \tilde{P}, f \rangle = \int_{\mathbb{R}^n} P(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}, \tag{75}$$

while the Dirac delta maps each function to its value at the origin:

$$\langle \delta, f \rangle = f(0). \tag{76}$$

Distributions in this sense should not be confused with probability or frequency distributions, distributions as defined in differential geometry, or any of the many other scientific uses of the term.

Many operations require more care in their application to distributions than to functions. While distributions can be added together and multiplied with real numbers or with infinitely differentiable functions, the product of two distributions, for example, is not well-defined. One of the most useful operations that can be performed on a distribution is taking its derivative. This operation is defined by moving the derivative onto the test function (with a sign change), and allows all distributions to be treated as infinitely differentiable with many of the properties of classical derivatives. This allows us to meaningfully use objects like the n^{th} derivative of the Dirac delta:

$$\langle \delta^{(n)}, f \rangle = (-1)^n \langle \delta, f^{(n)} \rangle = (-1)^n f^{(n)}(0). \tag{77}$$

In describing the properties of distributions, it is useful to classify them by the sets of test functions that they handle gracefully. There are many choices

of f that could lead equation (75) to violate our definition of a distribution, such as any complex-valued function. Typically, test functions are drawn from the space $D(\mathbb{R}^n)$, which is the set of infinitely-differentiable, real-valued, compactly-supported functions. A distribution must linearly map any member of this set to a real number. The space of distributions is called $D'(\mathbb{R}^n)$, as the dual space of $D(\mathbb{R}^n)$.

By considering larger sets of test functions, one can define smaller sets of distributions that still linearly map all allowed test functions to the reals. Two such classes are used in this paper. The first is the set of tempered distributions. The test function of a tempered distribution does not have to be compactly supported, but can be any rapidly-decreasing smooth function. The space of these test functions is called Schwartz space or $S(\mathbb{R}^n)$ and notably includes Gaussians and their derivatives. By its integrability and boundedness, the most general form of our aperture filter is a tempered distribution, and tempered distributions are closed under differentiation: $\tilde{\kappa}, \tilde{\kappa}_x, \tilde{\kappa}_y \in S'(\mathbb{R}^2)$.

A useful subset of the tempered distributions is the set of distributions with compact support. These distributions map any test function to zero if the support of that function excludes a certain compact region, called the support of the distribution. The Dirac delta is a classic example of a compactly-supported distribution, because any test function with $f(0) = 0$ is mapped to zero, so $\text{supp}(\delta) = \{0\}$. We require the distributions induced by our post-processing operation and the texture to have compact support: $\tilde{m}, \tilde{P} \in \mathcal{E}'(\mathbb{R}^2)$.

All of this is relevant because we want to rigorously specify what m and κ can be, without requiring them to be differentiable functions. Specifically, we want to know when the quantity $M[I] = m{*}k{*}P$ is well-defined. The convolution theorem, which states that convolution can be performed by multiplication of Fourier transforms, holds for:

1. two L^1 functions, producing another L^1 function.
2. a tempered distribution and a compactly-supported distribution, producing a tempered distribution.
3. a rapidly decreasing function with a tempered distribution, producing another tempered distribution.

The first of these describes the traditional use of the theorem, the second is the reason we require P compactly supported for general (tempered) κ in the theorem, and the third lets us drop this assumption of compactness on P (which is bounded and locally integrable, so $\tilde{P}$ is tempered) in the corollary after $m * \kappa$ is specified as a rapidly-decreasing function.

We also use Schwartz's Paley-Weiner theorem, which states that the Fourier transform of a compactly-supported distribution on the reals is an entire function. This is a very powerful result in complex analysis, for which we suggest [8] as a reference. Complex analysis extends analysis to functions on the complex numbers, creating alternate versions of familiar ideas from calculus on the reals. Several of these appear in our proof, particularly in claims 3 and 4.

Perhaps the most important of these concepts is the complex derivative. This is defined, just as on the reals, as the limit of the difference quotient, but it will exist in far fewer cases. Take, for example, the function $\Re(z)$, which returns the real part of its complex input z. Using the standard metaphor of $\mathbb{R}^2$ for $\mathbb{C}^1$, we could imagine this function as having perfectly well-defined partial derivatives: 1 along the real axis, 0 along the imaginary axis. However, because the derivative is a single limit, which must match from all directions of approach in order to exist, the function $\Re(z)$ is in fact nowhere complex differentiable.

As a result of this restrictive definition, differentiable functions are much rarer in complex analysis, and they have a number of remarkable properties. Functions that are complex differentiable in a neighborhood, called analytic or holomorphic functions, are, for example, infinitely differentiable everywhere the first derivative exists.

For complex functions that are holomorphic except at isolated points, there are three kinds of singularities that can occur: removable singularities, poles, and essential singularities. A removable singularity is like a patchable hole in the function — the function is not defined at the point, but it can be continuously extended to a function that is. A pole is a point at which the function goes to complex infinity (a quantity with infinite magnitude and indeterminate phase) but where the product of the function and some polynomial is holomorphic at that point. Anything more serious, like an oscillating discontinuity or a non-pole infinity, is called an essential singularity.

A holomorphic function with no singularities at any point other than infinity is called an entire function, and these are very special. They include polynomials, exponentials, trigonometric functions, and their sums, products, compositions, derivatives, and integrals. According to Liouville's theorem, any entire function whose magnitude is bounded must be constant, so any non-constant entire function must have a singularity at infinity. If this singularity is essential, the function is transcendental (e.g. sin and cos) and if it is a pole, the function is a polynomial. We use this restriction, along with Schwartz's Paley-Weiner theorem, to prove claim 4.

Appendix B: Alternate Derivations of the Focal Flow Constraint

Putting aside the question of uniqueness, the correctness of the focal flow constraint (17) is easily verified by setting $I = k * P$ with Gaussian k and simply taking the relevant derivatives. Here we provide two alternative confirmations that may provide additional intuition. One of these derivations is based on a truncated Taylor expansion, mirroring a common derivation for linearized optical flow. The other is based on sinusoidal textures, illustrated in Figure 1 and analyzed in section 3 for inherent sensitivity.

B.1 From Taylor Expansion

Following the well-known Taylor series derivation for differential optical flow, we can consider the difference in intensity at a pixel between a pair of images taken a time step Δt apart. We take advantage of the fact that the brightness of the underlying sharp texture does not change, but we must correct for the change in blur to process the images.

To do so, we assume Gaussian blur kernels k,

$$k(x, y, \sigma) = \frac{e^{-\frac{x^2+y^2}{2\Sigma^2\sigma^2}}}{2\pi\Sigma^2\sigma^2}, \tag{78}$$

and define a reblurring filter b that takes narrow Gaussians to wider Gaussians under spatial convolution:

$$k(x, y, \sigma_2) = b\,(x, y, \sigma_1, \sigma_2) * k(x, y, \sigma_1)\,. \tag{79}$$

This reblurring filter takes the form

$$b(x, y, \sigma_1, \sigma_2) = k(x, y, \sqrt{\sigma_2^2 - \sigma_1^2}). \tag{80}$$

The unchanging texture brightness constraint states that for an all-in-focus pinhole image P,

$$P(x + \Delta x, y + \Delta y, t + \Delta t) = P(x, y, t), \tag{81}$$

with features moving from (x, y) to $(x + \Delta x, y + \Delta y)$ on the image. We are free to convolve both sides of this constraint by a Gaussian, for example:

$$k(x, y, \sigma(t + \Delta t)) * P(x + \Delta x, y + \Delta y, t + \Delta t) = k(x, y, \sigma(t + \Delta t)) * P(x, y, t). \tag{82}$$

Then, for images blurred with different Gaussian kernels, where we set the sign of Δt without loss of generality so that $\sigma(t + \Delta t) > \sigma(t)$, we can express this modification of the unchanging texture brightness constraint in terms of blurred images I:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = b\left(x, y, \sigma(t), \frac{Z(t + \Delta t)}{Z(t)}\sigma(t + \Delta t)\right) * I(x, y, t), \tag{83}$$

where the $\frac{Z+\Delta Z}{Z}$ term accounts for the change in magnification between images. Taking the Taylor expansion of either side and dropping terms above first order, we have the approximation

$$I(x, y, t) + I_x\Delta x + I_y\Delta y + I_t\Delta t \;\approx\; \delta(x, y) * I(x, y, t) + \left(\Delta t\left(\dot{Z}\sigma/Z + \dot{\sigma}\right) b_{\sigma_2}(x, y, \sigma, \sigma) * I(x, y, t)\right). \tag{84}$$

Subtracting the $I(x, y, t)$ term from each side, dividing by Δt, and noting that

$$\left(\dot{Z}\sigma/Z + \dot{\sigma}\right) b_{\sigma_2}(x, y, \sigma, \sigma) = -\,v\,(b_{xx}(x, y, \sigma, \sigma) + b_{yy}(x, y, \sigma, \sigma)) \tag{85}$$

$$= -\,v(\delta_{xx} + \delta_{yy}), \tag{86}$$

our approximate constraint becomes

$$I_x\frac{\Delta x}{\Delta t} + I_y\frac{\Delta y}{\Delta t} + I_t \approx -\,v(I_{xx} + I_{yy}). \tag{87}$$

In the absence of blur, $v = 0$ and this is identical to optical flow. In the limit as Δt approaches zero, and under the separation of $(\dot{x}, \dot{y})$ into translation and magnification terms, this produces the focal flow constraint (17).

B.2 From Sinusoidal Textures

For general sinusoidal texture

$$T(a,b) = \sin(\omega_a a + \omega_b b + \phi_0) \tag{88}$$

a pinhole camera will record the image

$$P(x,y,t) = \sin(\omega_x(t)x + \omega_y(t)y + \phi(t)), \tag{89}$$

$$\omega_x = -\frac{Z(t)}{\mu_s}\omega_a, \tag{90}$$

$$\omega_y = -\frac{Z(t)}{\mu_s}\omega_b, \tag{91}$$

$$\phi = \omega_a X(t) + \omega_b Y(t) + \phi_0. \tag{92}$$

Under Gaussian blur as in equation (78), frequency and phase will not change but amplitude will:

$$I(x,y,t) = B(t)\sin(\omega_x x + \omega_y y + \phi), \tag{93}$$

$$B(t) = \max_{\phi}(k * P) = e^{-\Sigma^2(\omega_x^2+\omega_y^2)\sigma^2/2}. \tag{94}$$

The derivatives of this image are as follows:

$$I_x = \omega_x B\cos(\omega_x x + \omega_y y + \phi), \tag{95}$$

$$I_y = \omega_y B\cos(\omega_x x + \omega_y y + \phi), \tag{96}$$

$$I_{xx} = -\omega_x^2 B\sin(\omega_x x + \omega_y y + \phi), \tag{97}$$

$$I_{yy} = -\omega_y^2 B\sin(\omega_x x + \omega_y y + \phi), \tag{98}$$

$$\begin{aligned} I_t =& (\dot{\phi} + \dot{\omega_x}x + \dot{\omega_y}y)B\cos(\omega_x x + \omega_y y + \phi) \\ &+ \dot{B}\sin(\omega_x x + \omega_y y + \phi), \end{aligned} \tag{99}$$

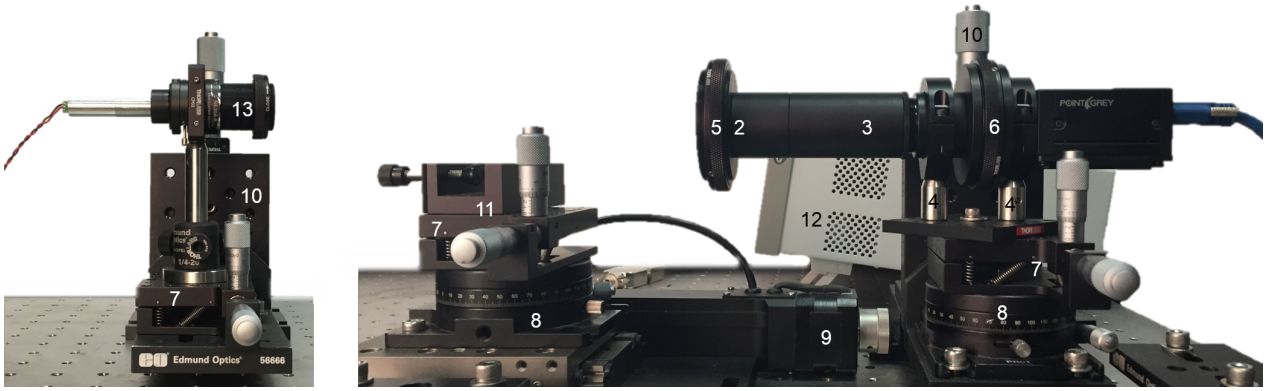
so that

$$\begin{aligned} I_t =& \frac{\omega_a\dot{X}}{\omega_x}I_x + \frac{\omega_b\dot{Y}}{\omega_y}I_y + \frac{\dot{\omega_x}x}{\omega_x}I_x + \frac{\dot{\omega_y}y}{\omega_y}I_y \\ &+ \frac{\dot{B}}{-B(\omega_x^2+\omega_y^2)}(I_{xx}+I_{yy}) \end{aligned} \tag{100}$$

$$= -u_1 I_x - u_2 I_y - u_3 x I_x - u_3 y I_y - v I_{xx} - v I_{yy}. \tag{101}$$

By the linearity of convolution and differentiation, equation (101) holds for all sum-of-sinusoid textures, so that the focal flow constraint applies to any texture with a Fourier transform.

Appendix C: List of Parts

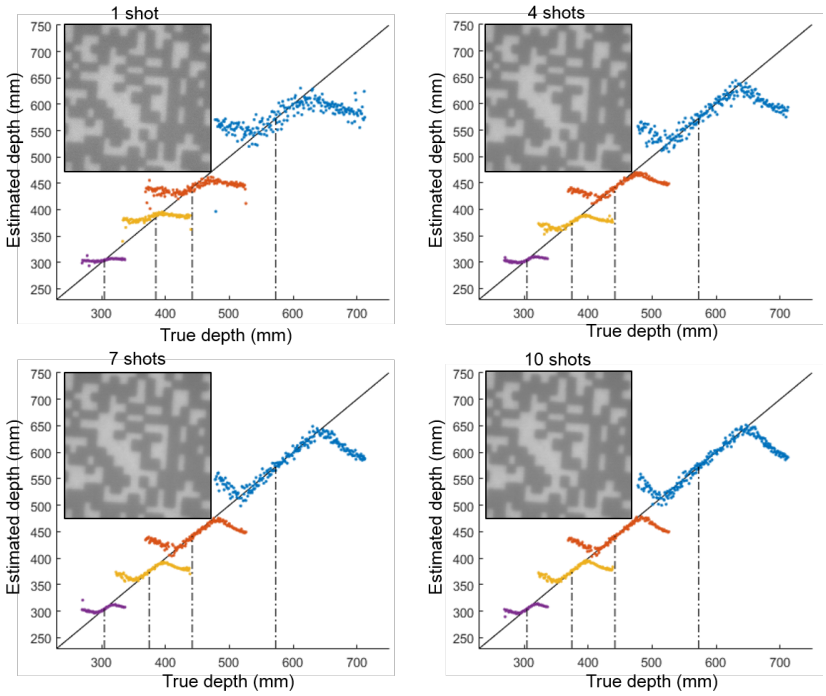


No.	Component	Source	Part Number	Quantity	Description
1	Camera	Point Grey	GS3-U3-23S6M-C	1	High speed, monochrome, powered by USB
2	Lens	Thorlabs	LA1509-A	1	Planar-convex, $\varnothing 1''$, $f = 100$mm, AR coated(350-700nm)
2	Apodizing Filter (Optional)	Thorlabs	NDYR20B	1	Reflective, $\varnothing$25mm, ND, OD: 0.04 - 2
3	Lens Tube	Thorlabs	SM1 Family	Flexible	SM1 thread, $\varnothing 1''$, recommend SM1V15 for adjustable μ_s
4	Lens Tube Mounts	Thorlabs	SM1TC+TR075	2	
5	Aperture Diaphragm	Thorlabs	SM2D25D or SM1D12D	1	SM1/SM2 thread, $\varnothing 2''$ or $\varnothing 1''$, removed when using apodizing filter
6	Calibration Diaphragm	Thorlabs	SM2D25D	1	SM2 thread, $\varnothing 2''$, connected with SM1A2 and SM2A6
7	Pitch & Yaw Platform	Thorlabs	PY003	3	
8	Rotation Platform	Thorlabs	PR01+PR01A	2	
9	Translation Stage	Thorlabs	LNR50S	1	Controlled and powered by 12
10	X-Y Translation Stage	Thorlabs & EO	2×PT1+PT101+ PT102+EO56666	2	
11	Wide Plate Holder	Thorlabs	FP02	1	
12	Stepper Motor Controllers	Thorlabs	BSC201	1	Powered by 110V, connected with PC via USB
13	Laser	Thorlabs	CPS532	1	Mounted with AD11F, SM1D12SZ, CP02, NE20A-A, SM1D12D

Appendix D: Detailed Experimental Results

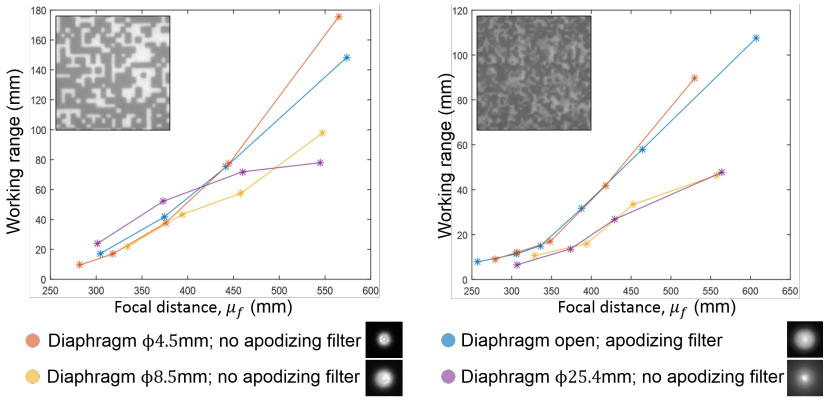
Performance versus noise

To counteract sensor noise, several shots can be averaged to create an input image (inset) to the measurement algorithm. (zoom in to see difference in noise level). Comparing measured depth to ground truth (solid black line) shows that that, as expected, measurement accuracy improves with shot count. Unless otherwise noted, all results the paper use 10-shot averages.



Working range versus aperture

We show working range ($\equiv$ range for which depth error $< 1\%\mu_f$) versus focal depth μ_f for four apertures, over two scene textures. We also show a sample point spread function for each aperture, at the same scale as the input image.



Performance for varying apertures and textures

Distance measurements versus ground truth (black lines) for a variety of focal distances and aperture configurations. Each row is a different aperture configuration, and the left and middle columns show results for both lower-frequency scene textures (left column) and higher-frequency scene textures (middle column). The right-most column shows corresponding sample point spread functions, each for a variety of depths. The measurement algorithm is quite robust to deviations from the idealized Gaussian blur model. From top to bottom, the aperture configurations are: (I) diaphragm ⌀4.5mm, no filter; (II) diaphragm open, with apodizing filter; (III) diaphragm ⌀8.5mm, no filter; (IV) diaphragm ⌀25.4mm, no filter.