Focal Track: Depth and Accommodation with Oscillating Lens Deformation

Qi Guo, Emma Alexander, and Todd Zickler
Harvard School of Engineering and Applied Science
qguo@seas.harvard.edu

Abstract

*The focal track sensor is a monocular and computationally efficient depth sensor that is based on defocus controlled by a liquid membrane lens. It synchronizes small lens oscillations with a photosensor to produce real-time depth maps by means of differential defocus, and it couples these oscillations with bigger lens deformations that adapt the defocus working range to track objects over large axial distances. To create the focal track sensor, we derive a texture-invariant family of equations that relate image derivatives to scene depth when a lens changes its focal length differentially. Based on these equations, we design a feed-forward sequence of computations that: robustly incorporates image derivatives at multiple scales; produces confidence maps along with depth; and can be trained end-to-end to mitigate against noise, aberrations, and other non-idealities. Our prototype with 1-inch optics produces depth and confidence maps at 100 frames per second over an axial range of more than 75*cm.

1. Introduction

Depth from defocus is an attractive ranging modality for micro-robots, wearables, and other small, low-power platforms because it fits into a small monocular package, does not require an active light source, and can be implemented in ways that produce depth maps with low computational power. This has previously been demonstrated with defocus that is controlled by manipulating a camera's aperture diaphragm [13], the distance between a sensor and lens [17], or a camera's position [1].

Here we describe a defocus range sensor that is instead based on lens deformation, which has the practical advantage of providing a large working range with little mechanical power. The working range of any depth from defocus system behaves like a depth of field in photography: once an object is too far in front or behind the in-focus plane, the depth-dependent contrast effects become too small to detect and analyze. The advantage of using lens deformation is that the sensor can easily mimic the accommodation reflex found in vertebrate vision, adaptively shifting its in-focus distance to match a moving object, and thereby extending its defocus working range. See Figure 1.

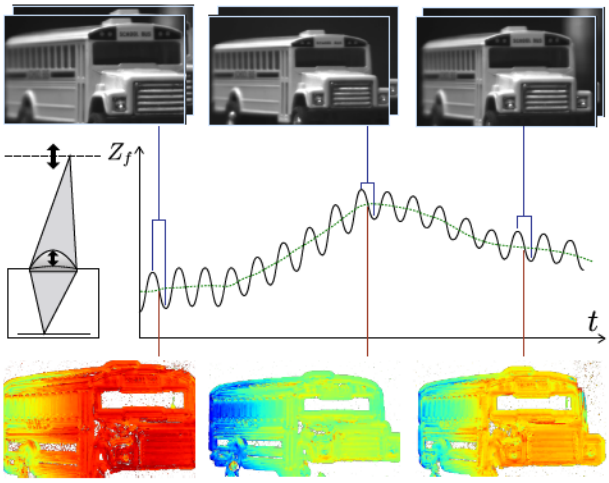

Figure 1: Focal tracking. A deformable lens creates exposure-synchronized oscillations of the in-focus distance Z_f, producing sequential pairs of video frames (top) with slightly different defocus. Depth and confidence maps (bottom) are computed from each pair, at rates up to 100fps. The depth also feeds back to the lens controller so that the system can "accommodate" by adjusting its in-focus distance (green line) to match a moving object.

We call this focal tracking, and we design our focal track sensor in two steps. First, we use an idealized optical model to derive a new texture-invariant family of equations that directly relate the spatial and temporal image derivatives at a pixel, $\nabla^2 I, I_t$, to the scene depth Z at the back-projection of that pixel. These equations are derived by analyzing differential defocus in the spirit of Alexander *et al.* [1], but with a different optical model that consists of a thin lens with Gaussian blur and time-varying focal length $f(t)$. The simplest equation in the derived family is

$$Z = \frac{\alpha \nabla^2 I}{\beta \nabla^2 I + I_t}, \tag{1}$$

where α, β are constants determined by the physical characteristics of the sensor and lens. The simplicity and texture invariance of these equations are what enable the rapid two-shot depth computation shown in Figure 1.

In the second design step, we construct a computational tree that produces reliable depth maps at high frame rates by using finite differences at multiple scales to robustly implement Equation 1 and its relatives. The computational tree is designed to produce confidence maps in addition to depth, which allows the automatic identification of problematic image locations such as textureless regions and depth discontinuities. Also, while the structure of our computational tree is closely tied to the physical model, it has the important property that all of its free parameters can be jointly optimized using end-to-end training with a loss function that measures the quality of the final depth and confidence maps. This eliminates the need for manual parameter tuning, and it allows the system to perform well even in the presence of non-Gaussian point spread functions, sensor noise, optical aberrations, and other non-idealities.

To validate the system, we build a prototype using an off-the-shelf sensor and deformable lens, along with custom control electronics. When running on a GPU, it provides useful depth and confidence maps at 100 frames per second. The code, trained model parameters, and hardware details are available at http://vision.seas.harvard.edu/focaltrack.

2. Related Work

The general idea of using lens deformation to sense depth is at least two decades old [13], but the required lens technology has only recently emerged. It is not the only way to achieve accommodation, but compared to other ways of adapting the in-focus position—moving the entire camera [1] or changing the sensor/lens separation [17]—it has the advantage of requiring less space and mechanical power.

Computationally, there is a healthy variety of techniques for depth from defocus. Among them, ours is more similar to the fast, non-iterative methods of Subbarao [12, 13, 14] and Watanabe and Nayar [17]; and less similar to iterative methods [8, 16, 7, 18, 3] that rely on statistical models of sharp natural images and are more geared toward digital refocusing applications [15, 6]. Our notion of an explicit confidence map seems new to depth from defocus, and it may be particularly useful for real-time sensing applications, like gesture-driven interfaces and robotic grasping, that can tolerate some missing (low-confidence) values in the depth map as long the reported high-confidence ones are accurate.

Our derivation draws inspiration from previous analyses of Gaussian defocus [12, 13, 4], and in particular from that of Alexander *et al*. [1], who analyzed differential changes in Gaussian defocus as is done here, but for a different optical configuration that has a standard lens and sensor moving as a fixed pair. Mathematically, that configuration has the advantage of measuring velocity in addition to depth, but the one in this paper has the advantage of producing measurements at every pixel instead of larger image patches.

The deformable lens that we use in our prototype has previously been used for focal sweep imaging [9]. It is capable of normal-mode oscillations up to about 100Hz. This places an upper bound on the frame rate of a focal track system, but is likely to increase in the future as deformable lens technology continues to evolve (*e.g*. [11]).

3. Physical Model

We consider a static scene imaged through a wide-aperture lens, so that the captured image I is the spatial convolution of a depth-scaled filter k with the sharp texture P that would have been captured by a pinhole camera:

$$I(x, y, t) = k(x, y, \sigma(t)) * P(x, y). \tag{2}$$

We assume Gaussian blur of RMS width $\sigma(t)\Sigma$:

$$k = \frac{1}{\sigma(t)^2} \exp\left(-\frac{x^2 + y^2}{2\sigma(t)^2\Sigma^2}\right). \tag{3}$$

Here, the constant Σ is the width of the kernel at uniform magnification, and the magnification factor $\sigma(t)$ follows the thin lens rule and is determined by: the separation μ_s between the sensor and lens; the lens' dioptric power ρ (for a lens with focal length f, $\rho = 1/f$); and the scene depth Z:

$$\sigma(t) = \left(\frac{1}{Z} - \rho(t)\right)\mu_s + 1. \tag{4}$$

We vary the dioptric power ρ of the lens over time using a deformable lens, and show that this enables a computationally efficient method for measuring depth.

3.1. Depth from Differential Focus Change

Under a differential change in dioptric power, the brightness of each pixel in the image will change as the scaled blur kernel re-weights the underlying scene texture:

$$I_t = k_t * P. \tag{5}$$

For Gaussian blur, $k_t \propto \nabla^2 k$, which implies that the same time and space derivatives of the *blurred image* are also proportional, regardless of the underlying texture P:

$$\nabla^2 I = \nabla^2 k * P = \frac{k_t}{\Sigma^2\sigma\dot{\sigma}} * P = \frac{1}{\Sigma^2\sigma\dot{\sigma}} I_t. \tag{6}$$

For more insight into the special relationship between Gaussian blur and texture-invariant depth sensing, see [1].

This proportionality of image derivatives provides a simple expression of object depth Z from image values and

knowable camera parameters:

$$Z = V/W, \tag{7}$$

$$V = (\Sigma^2 \mu_s^2 \dot{\rho}) \nabla^2 I, \tag{8}$$

$$W = (\Sigma^2 \mu_s \dot{\rho})(\mu_s \rho - 1) \nabla^2 I - I_t. \tag{9}$$

This is Equation 1 from the introduction, and as we noted there, it provides per-pixel depth from a monocular camera with simple, non-iterative computation.

We can generalize this equation to a larger family of equations that are all invariant to texture but can provide complimentary information about depth. To see how, note that Equation 7 is undefined when both V and W go to zero, which occurs in textureless regions and at the center of light-dark edges. All defocus cues fail in textureless regions, but edges can be handled by noting that the ratio $\frac{k_t}{\nabla^2 k} = \Sigma\sigma\dot{\sigma}$ is independent of image location, so that for any derivative orders $j, k \in \mathbb{N}$ and constants $\lambda_{jk} \in \mathbb{R}$, depth can also be recovered using

$$Z = \frac{\sum_{j,k} \lambda_{jk} \partial_x^j \partial_y^k V}{\sum_{j,k} \lambda_{jk} \partial_x^j \partial_y^k W}. \tag{10}$$

This means that at pixels where Equation 7 ($j, k = 0$) fails, we can still hope to measure depth using higher order derivatives. In practice, we find it useful to compute multiple depth estimates using a few independent instantiations of Equation 10 (*i.e.*, with V/W, V_x/W_x, and V_y/W_y) and combine them with adaptive weights. See Section 4.

3.2. Confidence

Regardless of how many derivatives are used, there will always be problematic image locations where the defocus cue fails. This includes textureless regions, regions beyond the working range, points near depth discontinuities, and so on. We handle these by computing a confidence value to accompany the depth estimate at every pixel. These general-purpose confidence maps are informed by the local image values and can be used in many ways, from simply discarding low-confidence estimates as we do in Section 6, to fusing the depths with other visual cues (*e.g.*, [2]).

To derive an expression for confidence, we use an additive zero-mean Gaussian noise model on the image, which propagates to means and variances of Gaussian random variables $\tilde{V}$ and $\tilde{W}$ in terms of physical camera parameters, derivative scale, and the assumed image noise level. Then the variance of the depth $\tilde{Z} = \tilde{V}/\tilde{W}$ can be expressed, to first order, as

$$\mathrm{Var}[\tilde{Z}] \approx \frac{\mathrm{E}[\tilde{V}]^2}{\mathrm{E}[\tilde{W}]^2} \left(\frac{\mathrm{Var}[\tilde{V}]}{\mathrm{E}[\tilde{V}]^2} + \frac{\mathrm{Var}[\tilde{W}]}{\mathrm{E}[\tilde{W}]^2} - \frac{2\mathrm{Cov}[\tilde{V}, \tilde{W}]}{\mathrm{E}[\tilde{V}]\mathrm{E}[\tilde{W}]} \right). \tag{11}$$

The confidence C in a given depth measurement is the inverse of this measurement, scaled to the range [0,1], so that for constants $\omega_0, \omega_1, \omega_2$ determined by the noise level, camera parameters, and derivative scale and order,

$$C = \left(\frac{\omega_0}{W^2} + \frac{V^2 \omega_1}{W^4} + \frac{V \omega_2}{W^3} + 1 \right)^{-1/2} \approx \left(\mathrm{Var}[\tilde{Z}] + 1 \right)^{-1/2}. \tag{12}$$

Here, we have used the measured values V and W as proxies for $\mathrm{E}[\tilde{V}]$ and $\mathrm{E}[\tilde{W}]$, which gives a parametric confidence model with three tuneable parameters.

Expressions follow immediately for depth measurements based on derivatives of $\tilde{V}$ and $\tilde{W}$. Note that the relative value of these expressions is determined by local image values, so that problem patches for each derivative order can be identified by their lower confidence values.

4. Computational Tree and Accomodation

A primary concern when implementing these calculations is choosing effective spatial derivative filters for measuring $\nabla^2 I$ and $\partial_x^j \partial_y^k \nabla^2 I$. These filters must include smoothing for noise suppression, and must also be normalized to avoid bias in Equation 10 and 12. Moreover, their preferred spatial support depends on the (unknown) scale of the features at each location of the underlying image.

Our approach is to use a set of Gaussian derivative filters $\partial_x^j \partial_y^k \nabla^2 G(i)$ with different sizes/scales i to create image pyramids $\partial_x^j \partial_y^k \nabla^2 I^i$. We compute separate depth and confidence estimates for each of these "channels", and then we combine them using image-dependent weights that are based on the estimated confidences. The channels include a variety of scales i so that the system has the flexibility to adapt to the feature size at each image location, and they also include a few different orders j, k to mitigate against the degeneracy described in Section 3.1.

To make this work, we design a feed-forward computational tree (Figure 2) that merges parallel depth and confidence estimates from each triple (i, j, k). Apart from being efficient, the most important property of this approach is that all of the parameters—including both the optical ones Σ, μ_s, ρ and computational ones $\{\omega\}$—can be end-to-end optimized by automatic differentiation and gradient descent on a loss function that measures the overall quality of the combined depth and confidence map.

End-to-end optimization avoids the limitations of manual parameter tuning. More importantly, it trains the confidence to predict the reliability of depth estimates from different (i, j, k) at each location by fine tuning $\{\omega\}$. Furthermore, it provides a principled way of handling non-Gaussian point spread functions that are unavoidable when using real lenses. Instead of trying to measure the point spread functions and approximate them with values of (Σ, μ_s, ρ), the end-to-end approach automatically finds values that are optimal for the task at hand: producing accurate, high-confidence depth at the image locations that allow it.

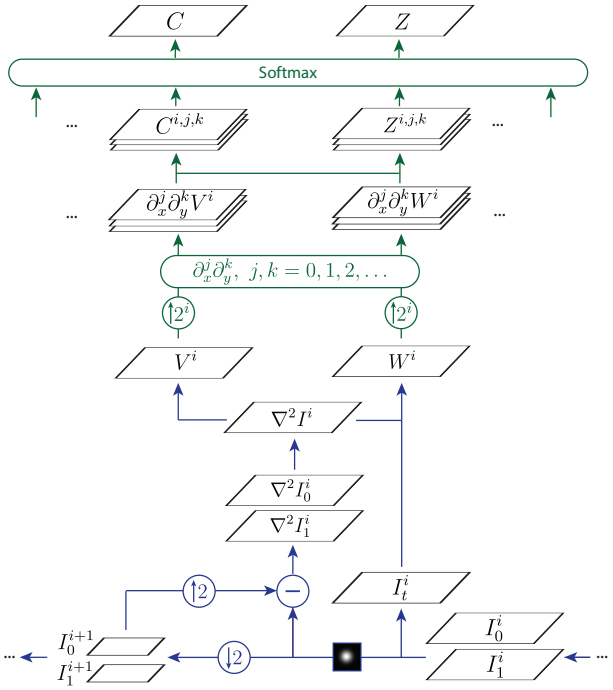

Figure 2: Computational tree. Lower nodes compute conventional Gaussian and Laplacian pyramids for each input pairs (I_0, I_1), to obtain intermediate maps (V^i, W^i) at each scale i. Upper nodes apply a set of derivative filters $\{\partial_x^j\partial_y^k\}$, and estimates per-$ijk$ depth and confidence maps that are ultimately fused via softmax.

The input to the tree is a pair of images (I_0, I_1) captured with known and slightly different dioptric powers $(\rho + \Delta\rho, \rho - \Delta\rho)$. Each is decomposed into pyramids, and for each scale i, we compute a temporal finite difference $I_t^i = 0.5(I_0^i - I_1^i)$, the spatial Laplacian $\nabla^2 I^i = 0.5(\nabla^2 I_0^i + \nabla^2 I_1^i)$, and the intermediate maps

$$(V^i, W^i) = (a^i b \nabla^2 I^i, a^i \nabla^2 I^i - I_t). \tag{13}$$

The single parameter b models the optical term $\mu_s/(\mu_s\rho - 1) = Z_f$, which is the average in-focus distance for the two images, and the per-scale parameter a^i models the combination of the optical term $(\Sigma^2\mu_s\dot{\rho})(\mu_s\rho - 1)$ and the normalization factor induced by the filters at scale i.

Each 2D map (V^i, W^i) is bilinearly upsampled to full resolution and spatially filtered with a difference filter for each (j, k). Depth and confidence maps are computed via

$$Z^{i,j,k} = \frac{\partial_x^j\partial_y^k V^i}{\partial_x^j\partial_y^k W^i} + c, \tag{14}$$

$$C^{i,j,k} = \left(\frac{\omega_0^{i,j,k}}{\left(\partial_x^j\partial_y^k W^i\right)^2} + \frac{\omega_1^{i,j,k}\left(\partial_x^j\partial_y^k V^i\right)^2}{\left(\partial_x^j\partial_y^k W^i\right)^4} + \frac{\omega_2^{i,j,k}\left(\partial_x^j\partial_y^k V^i\right)}{\left(\partial_x^j\partial_y^k W^i\right)^3} + 1 \right)^{-1/2}, \tag{15}$$

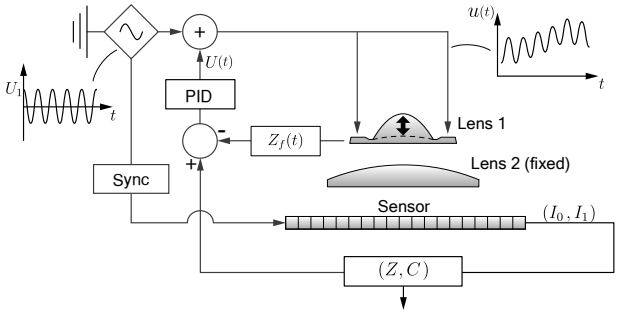

Figure 3: Feedback control for accommodation. The median of the measured scene depth feeds back to the lens controller, as a slow-changing offset to the sinusoidal control signal. This keeps moving objects in focus, where the defocus cue is strongest, for an extended working range.

where parameter c models the sensor's position in some global coordinate system. Finally, the per-ijk maps are fused via softmax:

$$(Z, C) = \sum_{i,j,k} \beta^{i,j,k}(Z^{i,j,k}, C^{i,j,k}),$$
$$\beta^{i,j,k} = \frac{\exp(C^{i,j,k})}{\sum_{i,j,k}\exp(C^{i,j,k})}. \tag{16}$$

As shown in Figure 3, we implement accommodation using a PID controller to add offsets to the focal oscillations in order to match the median of the high-confidence depth estimates across the scene, or a region of interest within it. The lens signal is a sinusoid $u(t) = U(t) + U_1\cos(\omega t)$ with constant U_1, frequency ω that matches the sensor's frame rate, and control signal $U(t)$. The image pairs are captured at $U(t) + U_1$ and $U(t) - U_1$, and the dioptric power is linear to the signal, $\rho(t) = \eta U(t)$.

5. Calibration

We capture calibration data using scenes that are planar and normal to the optical axis, so that the ground truth depth map Z^* is constant for each frame. Each calibration frame includes Z^*, a pair of images (I_0, I_1), and the dioptric powers $(\rho - \Delta\rho, \rho + \Delta\rho)$ under which they were captured.

The tunable parameters are $\{a^i, b, c, \omega^{i,j,k}\}$, and their preferred values depend on the sensor's (typically, non-Gaussian) point spread functions. For this reason, the dioptric difference $2\Delta\rho$ remains fixed during calibration and then during subsequent use. During accommodation, a sensor's point spread functions change, and so do the preferred parameter values. We handle this by calibrating the parameters at a single dioptric power offset $\rho = \rho_0$ to obtain $\{a^i, b, c, \omega^{i,j,k}\}(\rho_0)$ and then adapting them according to the known $\rho(t)$ during accommodation.

To derive adaptation equations, recall our physical model for in-focus distance:

$$Z_f(\rho(t)) = \frac{(\mu_s\rho_0 - 1)Z_f(\rho_0)}{\mu_s\rho(t) - 1} = \frac{(\mu_s\rho_0 - 1)Z_f(\rho_0)}{\mu_s\eta U(t) - 1}. \quad (17)$$

This immediately suggests the adaptive model for b:

$$b(\rho(t)) = (d_1 U(t) - d_2)^{-1}, \quad (18)$$

with constants d_1 and d_2. Values for the two constants are determined by placing a planar patch at several known distances throughout the accommodation range, and for each one adjusting the lens signal U until the patch is in focus. This gives a set of (Z_f, U)-pairs that we use as a proxy for a set of (b, U)-pairs from which we can linearly fit d_1 and d_2. Then, the optical parameters $\{a^i\}$ adapt as

$$a^i(\rho(t)) = a^i(\rho_0)b(\rho_0)/b(\rho(t)). \quad (19)$$

The parameter c (depth offset) and confidence parameters $\{\omega^{i,j,k}\}$ are used for all lens settings without adaptation. The rest of this section describes calibration at ρ_0.

5.1. Initialization

The non-linearity of our model makes it important to obtain good initialization of parameters prior to gradient descent. We find it sufficient to initialize confidence parameters $\{\omega^{i,j,k}\}$ to 1, and good initialization for the other parameters can be derived from their physical interpretations.

The parameters b (in-focus distance) and c (depth offset) are simply initialized with crude values from a tape measure. The parameters $\{a^i\}$ are harder to measure directly, so we initialize them instead by fitting to the entire training set. For each scale i, we form the linear system

$$a^i \nabla^2 I^i = \frac{Z^* - c}{(Z^* - c) - b} I_t^i, \quad (20)$$

and set each a^i to its least-squares estimate:

$$a^i = \frac{(Z^* - c)\left(I_t^i \cdot \nabla^2 I^i\right)}{(Z^* - c - b)\left(\nabla^2 I^i \cdot \nabla^2 I^i\right)}. \quad (21)$$

5.2. Optimization

We jointly optimize the parameters by gradient descent, with the goal of minimizing mean depth error and maximizing number of high confidence predictions over the calibration dataset. At least two types of differentiable loss functions can be used. One type is the standard p-norm loss,

$$L(Z - Z^*, C) = \left(\text{mean}\left(|Z - Z^*|^p\right)\right)^{1/p}. \quad (22)$$

Another type of differentiable loss can be created using the recent notion of area under the sparsification curve

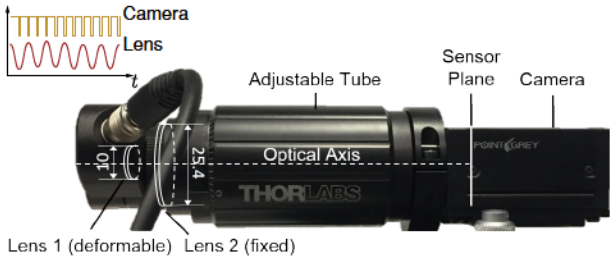

Figure 4: Sensor prototype. Exposure is synchronized with lens oscillations using signals shown top-left. All units mm.

(AUSC) [10]. A sparsification curve summarizes the accuracy and density of a confidence-weighted depth map for all possible thresholds of confidence, and good performance means small area under this curve. We can use AUSC directly as a loss function:

$$L(Z - Z^*) = \frac{1}{M} \sum_{\text{samples}} \lambda \cdot \text{Sort}(|Z - Z^*|, C), \quad (23)$$

where M is the number of training samples, $\lambda = [\frac{1}{N}, \frac{1}{N-1} + \frac{1}{N}, \ldots, 1 + \ldots + \frac{1}{N}]$ is a constant vector, and $\text{Sort}(\cdot, C)$ arranges each element of a depth map according to its confidence in ascending order. The $p-$norm loss can be interpreted as optimizing the single point on the sparsification curve that shows the error of full density, rather than the area underneath it.

6. Prototype and Experiments

Our prototype is shown in Figure 4. It uses a single deformable lens (EL-10-30-C-VIS-LD-MV, Optotune) that supports dioptric powers $\rho_1 \in [-1.5\text{m}^{-1}, 3.5\text{m}^{-1}]$. It is coupled in close proximity to a second lens with power $\rho_2 = 10\text{m}^{-1}$, giving a combined dioptric power $\rho = \rho_1 + \rho_2 - \delta\rho_1\rho_2 \approx \rho_1 + \rho_2$ that we take to be linear with respect to the lens control signal. The lens assembly is coupled to a monochrome camera (GS3-U3-23S6M-C, Point Grey Research). The lens and camera are driven by a custom signal generator that produces two synchronized signals as depicted in the inset of Figure 4. Thus, each sequential pair of frames represents images I_0, I_1 that are captured with a slight change in dioptric power $(\rho - \Delta\rho, \rho + \Delta\rho)$ and can be fed directly into the computational tree.

To create calibration and validation data, we print ten diverse natural textures with a standard laser printer and mount them onto rigid planes that can be carefully positioned normal to the sensor's optical axis and at several known depths (relative to a fixed global reference point) that span the intended working range. For each texture and depth, we capture two 300×480 images I_0, I_1. This gives a set of triples of images and depths $\{(I_0, I_1, Z^*)\}$. We use

the data from two of the textures for calibration, and the remaining eight for validation.

We use a computational tree with four scales $i = 0, 1, 2, 3$ and zero-order and first-order derivatives $j + k = 0, 1$. Thus, there are twelve depth and confidence maps that are fused at the final layer. Optimization is implemented using TensorFlow to take advantage of its automatic differentiation functionality. Small constants are added to the denominators when implementing Equations 14 and 15 to avoid numerical instabilities. We optimize confidence parameters $\{\omega^{i,j,k}\}$ (with the constraint $\omega^{i,j,k} = \omega^{i,k,j}$) before optimizing the optical parameters $\{a^i, b, c\}$.

We compare the quality of the calibrations obtained using four different loss functions. In each case, optimization was executed using an NVIDIA Quadro K5000 GPU and Intel Xeon CPU E5620 x16 Processor. Experimentally we find the 1-norm loss requires about half the number of iterations to converge, making it roughly twice as fast as the other loss functions for calibration. Figure 5 compares the performance of these optimized calibrations, as well as the initialized one, on the validation dataset. In all four cases, optimization decreases error and expands working range, which we define as the range over which mean depth error on the validation set is 10% or less. Since the 1-norm loss has the best performance and also runs more quickly, we use this calibration for the remaining results below. Additional calibration details, including differentiation of the AUSC loss and the optimization times, are in a supplement [5].

6.1. Results with natural scenes

Figure 6 shows a typical pair of input frames I_0, I_1 and the output depth maps, thresholded at confidence 0.99. The figure shows both the final fused depth map and the internal depth maps that are produced in each scale/derivative channel. In every channel of the tree, the blurry background is deemed to be low confidence and is automatically rejected. Even inside the object, the channels corresponding to the finest scales ($i = 0$) are quite noisy and many of the depth estimates are discarded at this confidence level. The boxes overlaid at scale $i = 2$ in the figure highlight how channels corresponding to different derivatives can provide complementary information, with low-confidence areas in one channel often showing high confidence in another.

Figure 6 shows how depth maps in coarser channels tend to "bleed" beyond object boundaries, because of the simple upsampling that is used for fusion in our tree. This could be improved through a more sophisticated fusion scheme, but for the purposes of this paper, we simply discard the coarsest channel $i = 3$ for the remaining results.

Figure 7 shows fused depth maps for a collection of scenes with a variety of textures and textureless regions as well as depth discontinuities and areas of high curvature. Note that the sensor interprets highlights, shadows,

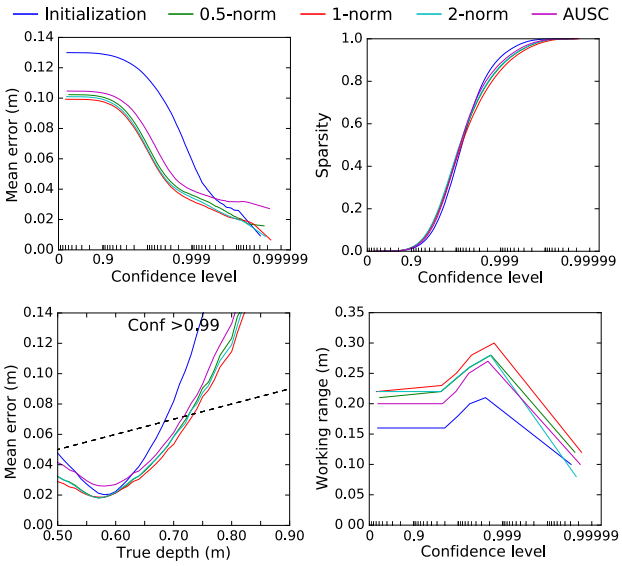

Figure 5: Calibration performance on validation set for both initialized parameters (blue) and optimized ones. Top row: Mean depth error and mean sparsity (*i.e.*, fraction of pixels without depth predictions) at increasing confidence thresholds. Bottom left: Mean depth error at different depths, with working range (dashed) defined as set of depths having less than 10% error. Bottom right: Working range versus confidence level. It initially expands with confidence level as the mean error decreases, and then it shrinks as blurry, low-contrast measurements at the edges of the working range are thresholded out. As shown in Figure 8, the latter effect can be avoided by accommodation.

and other lighting effects as additional textural structures on an object's surface. Figure 7**E** shows an example where a sheet of shiny, translucent bubble wrap is in the foreground.

Figure 8 demonstrates the benefit of accommodation. A toy leopard is moved through an axial distance of more than 40cm and observed with and without accommodation. With accommodation, depth is well-measured over a range of 50cm, while the other quickly degrade beyond 10cm from its fixed focal plane. The supplement [5] contains a more detailed evaluation of accommodation, showing that it increases the working range to more than 75cm.

The prototype, including accommodation, runs at 100fps and produces 300×480 depth and confidence maps using a laptop with NVIDIA GeForce GTX 1080 notebook graphics card and Intel Core i7 6820HK processor. Demo videos and code can be found on the project website.

Acknowledgement We thank Alex Kamovich for helpful discussions. This project was funded by US National Science Foundation awards IIS-1212928 and IIS-1718012, as well as NSF Graduate Research Fellowship No. DGE1144152 to E.A.

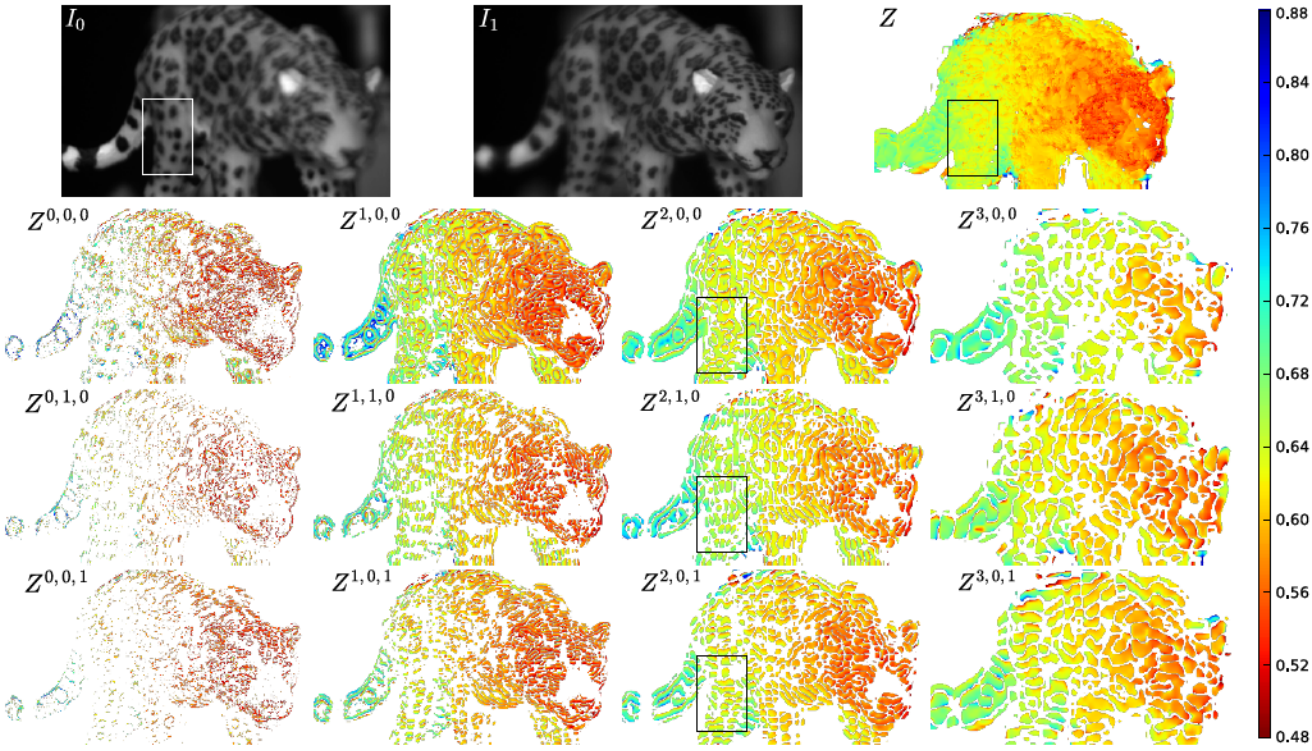

Figure 6: Depth map in meters of a toy leopard at confidence 0.99. The actual length of the toy is 13cm. Top row shows the image pair and fused depth map Z. Below are depth maps $Z^{i,j,k}$ for each channel of the computational tree. Overlaid boxes highlight how channels can fail at complimentary locations due to degeneracy, and how fusion can successfully merge them.

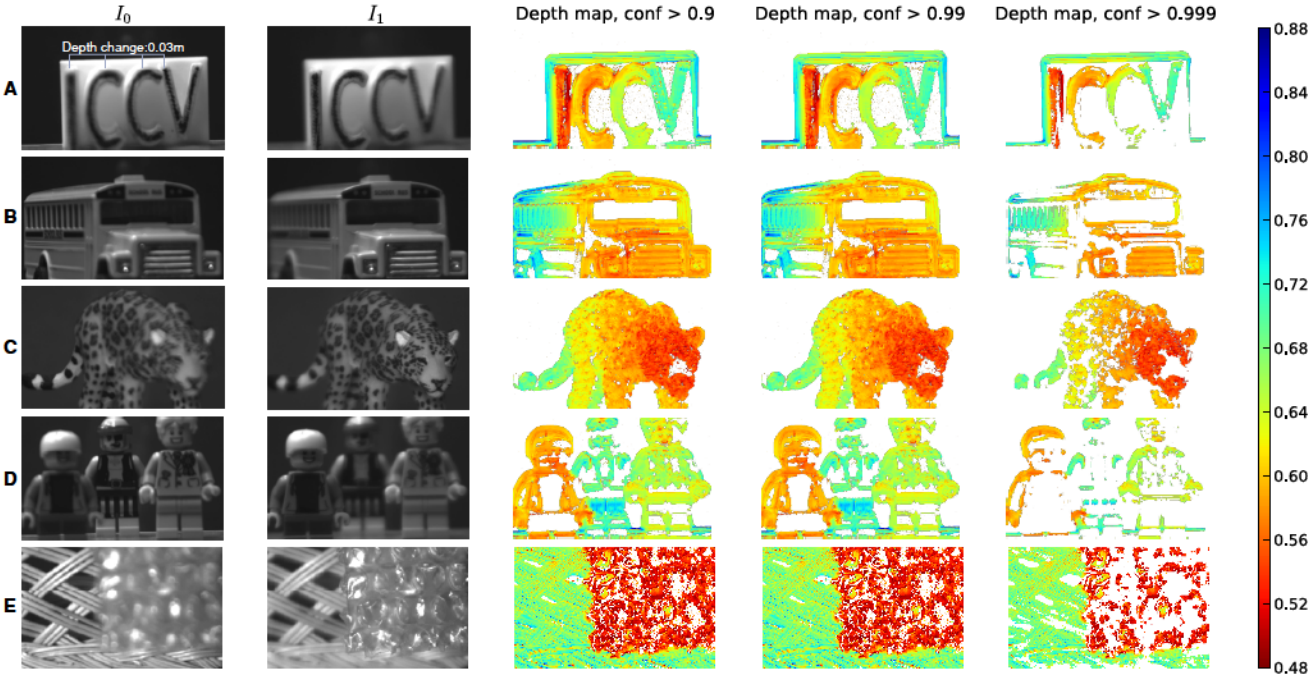

Figure 7: Measurements of several simple scenes with varying confidence thresholds. Depth in m.

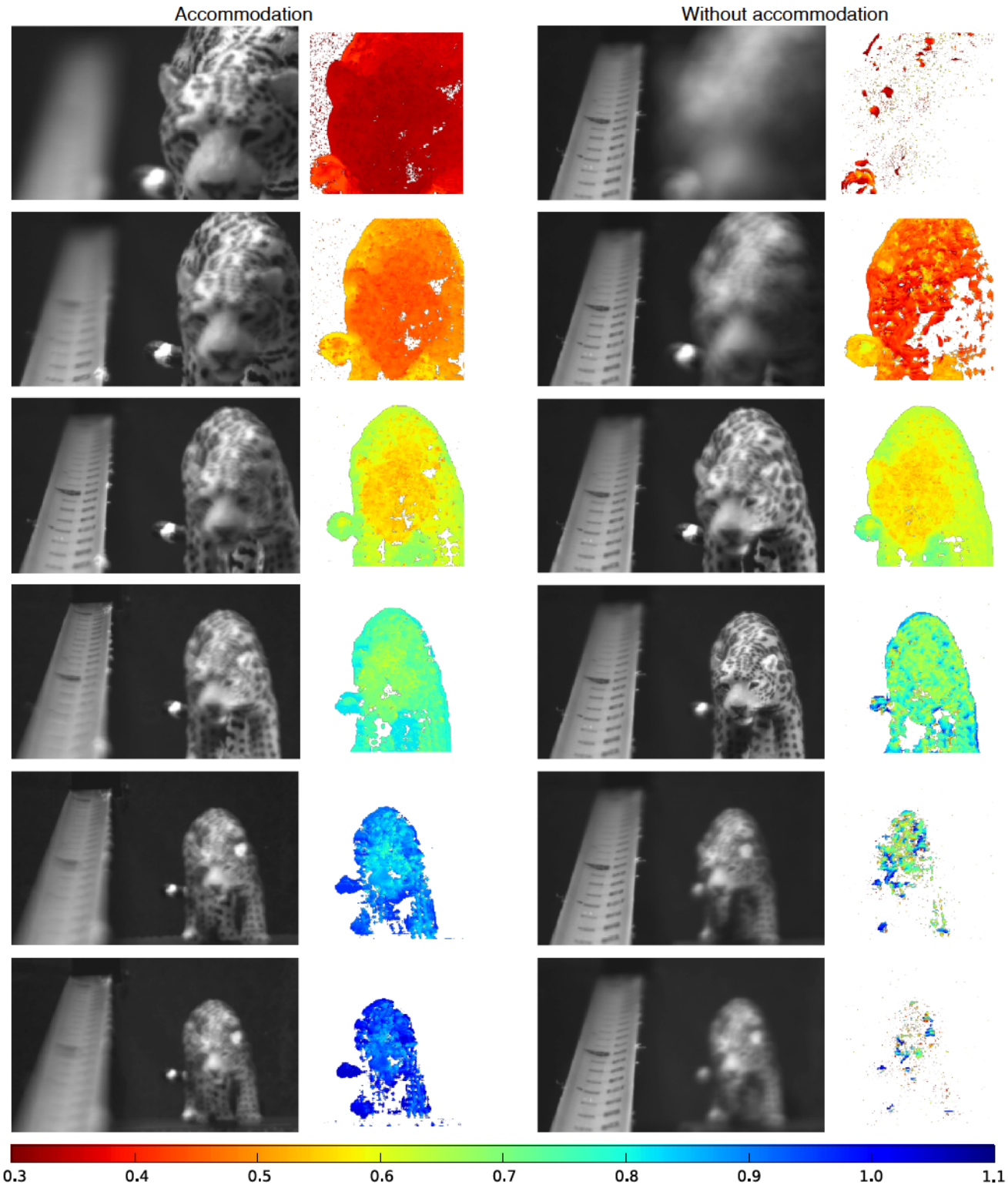

Figure 8: Demonstration with (left) and without (right) accommodation for confidence > 0.995 and depth in meters. The accommodation controller uses a region of interest that contains the toy but not the tape measure, which is only included to visualize the adapting focal depth (please zoom in). Accommodation more than doubles the useful range of the sensor.

References

[1] E. Alexander, Q. Guo, S. Koppal, S. Gortler, and T. Zickler. Focal flow: Measuring distance and velocity with defocus and differential motion. In *European Conference on Computer Vision*, pages 667–682. Springer, 2016. 1, 2

[2] J. T. Barron and B. Poole. The fast bilateral solver. In *European Conference on Computer Vision*, 2016. 3

[3] A. Chakrabarti and T. Zickler. Depth and deblurring from a spectrally-varying depth-of-field. In *European Conference on Computer Vision (ECCV)*, 2012. 2

[4] H. Farid and E. P. Simoncelli. Range estimation by optical differentiation. *Journal of the Optical Society of America A*, 15(7):1777–1786, 1998. 2

[5] Q. Guo, E. Alexander, and T. Zickler. Focal track: Supporting material. Technical Report TR-01-17, School of Engineering and Applied Science, Harvard University, 2017. 6

[6] H. Kim, C. Richardt, and C. Theobalt. Video depth-from-defocus. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 370–379. IEEE, 2016. 2

[7] A. Levin. Analyzing depth from coded aperture sets. In *European Conference on Computer Vision (ECCV)*, 2010. 2

[8] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM transactions on graphics (TOG)*, 26(3):70, 2007. 2

[9] D. Miau, O. Cossairt, and S. K. Nayar. Focal sweep videography with deformable optics. In *IEEE International Conference on Computational Photography*, 2013. 2

[10] A. Seki and M. Pollefeys. Patch based confidence prediction for dense disparity map. In *British Machine Vision Conference (BMVC)*, volume 10, 2016. 5

[11] C. A. Stan. Liquid optics: Oscillating lenses focus fast. *Nature Photonics*, 2(10):595, 2008. 2

[12] M. Subbarao. Parallel depth recovery by changing camera parameters. In *ICCV*, pages 149–155, 1988. 2

[13] M. Subbarao and G. Surya. Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294, 1994. 1, 2

[14] M. Subbarao and G. Surya. Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294, 1994. 2

[15] S. Suwajanakorn, C. Hernandez, and S. M. Seitz. Depth from focus with your mobile phone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3497–3506, 2015. 2

[16] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, and J. Tumblin. Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. In *ACM Transactions on Graphics (TOG)*, 2007. 2

[17] M. Watanabe and S. K. Nayar. Rational filters for passive depth from defocus. *International Journal of Computer Vision*, 27(3):203–225, 1998. 1, 2

[18] C. Zhou, S. Lin, and S. K. Nayar. Coded aperture pairs for depth from defocus and defocus deblurring. *International Journal of computer vision*, 93(1):53–72, 2011. 2