SpiderCam: Low-Power Snapshot Depth from Differential Defocus

Marcos A. Ferreira*,1 Tianao Li*,1 John Mamish*,2
Josiah Hester[2] Yaman Sangar[2] Qi Guo[3] Emma Alexander[1]
[1]Northwestern University [2]Georgia Institute of Technology [3]Purdue University
*Equal contributions, https://nubivlab.github.io/SpiderCam

Abstract

We introduce SpiderCam, an FPGA-based snapshot depth-from-defocus camera which produces 480×400 sparse depth maps in real-time at 32.5 FPS over a working range of 52 cm while consuming 624 mW of power in total. SpiderCam comprises a custom camera that simultaneously captures two differently focused images of the same scene, processed with a SystemVerilog implementation of depth from differential defocus (DfDD) on a low-power FPGA. To achieve state-of-the-art power consumption, we present algorithmic improvements to DfDD that overcome challenges caused by low-power sensors, and design a memory-local implementation for streaming depth computation on a device that is too small to store even a single image pair. We report the first sub-Watt total power measurement for passive FPGA-based 3D cameras in the literature.

1. Introduction

Jumping spiders accurately estimate distances using a brain the size of a poppy seed. Their behavior suggests that they use depth from defocus [49], aided by an eye that captures pairs of differently-defocused images simultaneously across layered retinas [33]. Inspired by their example, depth from differential defocus (DfDD) is a computationally efficient class of depth from defocus (DfD) algorithms that leverages small changes in defocus [1]. While DfDD has been shown to be more efficient than other DfD approaches in terms of floating point operations (FLOPs) [41], it has not previously been demonstrated in a real-world ultra-low-power system.

Low power depth sensing has instead been dominated by stereo, with the most efficient FPGA-based prototypes demonstrating 2-3 W of overall power consumption [45, 55, 65, 66]. Significantly, many existing works don't measure real-world power consumption or report quantitative real-world depth results (see Tab. 1). Instead, they benchmark on high-quality datasets like KITTI [11–13, 46] and Middlebury [25, 59–62], and estimate the "core power" of their algorithm alone. This ignores the power already expended through high-quality image sensors, I/O, and image alignment. It also likely overestimates depth accuracy for highly constrained systems, which may suffer from effects like sensor noise, miscalibration, and small baselines.

We show, for the first time, a complete FPGA-based 3D camera system operating under 1 Watt, and demonstrate a working range of over half a meter on real scenes measured on-device. Our camera captures simultaneous image pairs through a beam splitter with sensors at different offsets, like the eye of the jumping spider. In real time, our device outputs sparse depth maps, filtered by measurement confidence. Importantly, we demonstrate both the real-world total system power (including both sensors) and the real-world working range, on data with nonidealities from a compact, snapshot, low-power camera.

The algorithmic challenges presented by our tight power budget are threefold. First, the total operations are limited, even compared to previous DfDD methods, and we can no longer ignore the relative cost of different floating point operations (e.g., division can be 10× more expensive than multiplication). Second, low-power FPGAs impose tight resource constraints, requiring a memory-local streaming approach with a limited spatial footprint. Third, low-power sensors are small and noisy, requiring us to develop an algorithm with improved robustness to both optical aberrations and numerical errors. By combining a spider-like camera design for compact simultaneous image capture, with a novel algorithm tailored to our low-power electronics, SpiderCam demonstrates the promise of defocus for real-world low-power depth sensing.

2. Related Work

Depth imaging has been a major area of imaging research for decades, with solutions spanning the accuracy, speed, and SWaP-C (size, weight, power, and cost) tradeoff space. Broadly speaking, depth imaging can be separated into two categories: active and passive. Active techniques use a controlled light source to improve signal quality [18, 26], like LiDAR [7, 14] and structured light [14, 48, 51]. Reducing the SWaP-C of active techniques is challenging due to their

Name	Type	FPGA	Resolution	Core Power (W)		Real-World Quantitative Results	
				Min	Max	On-Device Working Range	Full System Power
Jin, 2014 [29]	Stereo	Kintex-6	640×480	1.93	2.67	none	none
Raj, 2014 [30]	DfD	Virtex-4†	400×400	0.46	0.58	0.77–0.80 m (**0.5**% **err**)	2 W + camera
Mattoccia, 2015 [45]	Stereo	Spartan-7	640×480	0.44	0.68	qualitative	2.5 W
Ttofis, 2015 [65]	Stereo	Kintex-7	1280×720	0.92	1.53	qualitative	2.8 W
Puglia, 2017 [55]	Stereo	Zynq Artix	1024×768	0.43	0.68	none	2 W
Zhang, 2018 [69]	Stereo	Kintex-7	1920×1080	0.89	1.23	none	none
Lu, 2021 [40]	Stereo	Kintex-7	1024×480	1.39	2.30	none	none
Wang, 2022 [66]	Stereo	Zynq US+	1920×1080	2.03	2.20	none	2.8 W + camera
Ours, normalized	DfD	Kintex-7	512×480	**0.42**	**0.55**	-	-
Ours, actual	DfD	ECP5	480×400	**0.24**	**0.31**	0.45–0.97 m (10% err)	**0.6 W**
Luo, 2025 [42]	DfD	N/A	480×360	N/A	N/A	**0.40–1.20 m** (10% err)	4.9 W

Table 1. **Power consumption for passive FPGA-based depth imaging systems.** "Core Power" numbers reflect only power consumed by logic for depth computation on aligned images; I/O and sensor power are ignored. Power estimates are derived using vendor tools and data from the corresponding works, normalized to a processing rate of 30 Mpix/sec across methods, see Sec. S8 for details. In addition to SOTA power efficiency in core power estimates, we are one of the very few systems to report full system power consumption and real-world depth performance (from noisy sensor captures rather than on existing datasets collected with large-baseline, high-power, pre-aligned image pairs). Note that we report two estimates for our method: a "normalized" estimate that handles more pixels per second on a less efficient FPGA for fairer comparisons with the literature, and a lower estimate reflecting the true parameters of our system. Additionally, we include Focal Split (Luo, 2025) [42], which reports a similar method on a Raspberry Pi 5 with real-world depth and power performance.
† This work uses an Xilinx Virtex-2 FPGA, but due to vendor tool unavailability, we estimate power for the more efficient Virtex-4.

need for illumination sources [18, 28, 37, 57]. Passive depth imaging systems, on the other hand, calculate depth images without using special hardware to illuminate the scene, and are the focus of this paper.

2.1. Monocular Depth Imaging

Estimating depth from a single monocular image is an ill-posed problem and cannot be solved without extensive reliance on priors [8, 47]. In spite of this, dense, high-accuracy results that run in real-time have been achieved by using deep learning to leverage large datasets. Although successful, these methods are ill-suited to space-constrained mobile computing: they rely on deep neural networks whose size [47] leads to computation and memory access energy costs which prevent the miniaturization of systems that use them, even with state-of-the-art edge TPUs [16, 50, 64]. If motion is introduced and a sequence of frames is used for the monocular depth imaging problem, it is no longer ill-posed and a broader class of both classical and deep learning algorithms is admitted [44, 52]. Although these methods yield high-quality results, their adoption in mobile computing systems is limited due to their compute and memory requirements [58, 63].

2.2. Depth from Stereo

Stereo vision is a passive depth imaging technique which uses two simultaneously captured images of the same scene with two different cameras spaced some distance apart [6]. By identifying translational differences in scene features between the two images. Stereo depth imaging systems are primarily concerned with identifying stereo matches [21]. Stereo matching is extensively researched, with both classical and deep-learning methods attracting significant attention [21, 32, 54]. Although deep-learning based stereo vision systems have achieved good results [54, 71], their significant compute and memory requirements rule out their appropriateness for space and power constrained systems. [71] compares 12 deep learning stereo vision methods, all of which run on Nvidia graphics cards (consuming upwards of 100 Watts [39]) and most of which run at only a few frames per second.

Hardware accelerators for stereo matching. Classical methods for stereo matching involve evaluating the cost of different pixel matches - a patch of pixels associated with a region is compared with patches in other regions and a minimal-cost matching is determined for each pixel in the image. While searching the entire image typically gives the best stereo matching results, to be computationally tractable, most methods only perform a restricted local search [21]. To make stereo depth cameras as compact and low-power as possible, significant work has been done in designing digital hardware to accelerate local stereo matching using FPGAs or ASICs.

Most hardware stereo matching accelerators work off of the same fundamental principle - two images are streamed in at the same time from two image sensors. Part of the image is stored on-chip in the FPGA and a census transform is used to implement semi-global matching [24, 40]. Ttofis *et al*. pre-filters images using edge filters to make the matching task easier [65]. Zhang *et al*. reduces hardware overhead by using alternate transformations for cost calculation [69]. Puglia *et al*. uses a dynamic programming inspired algo-

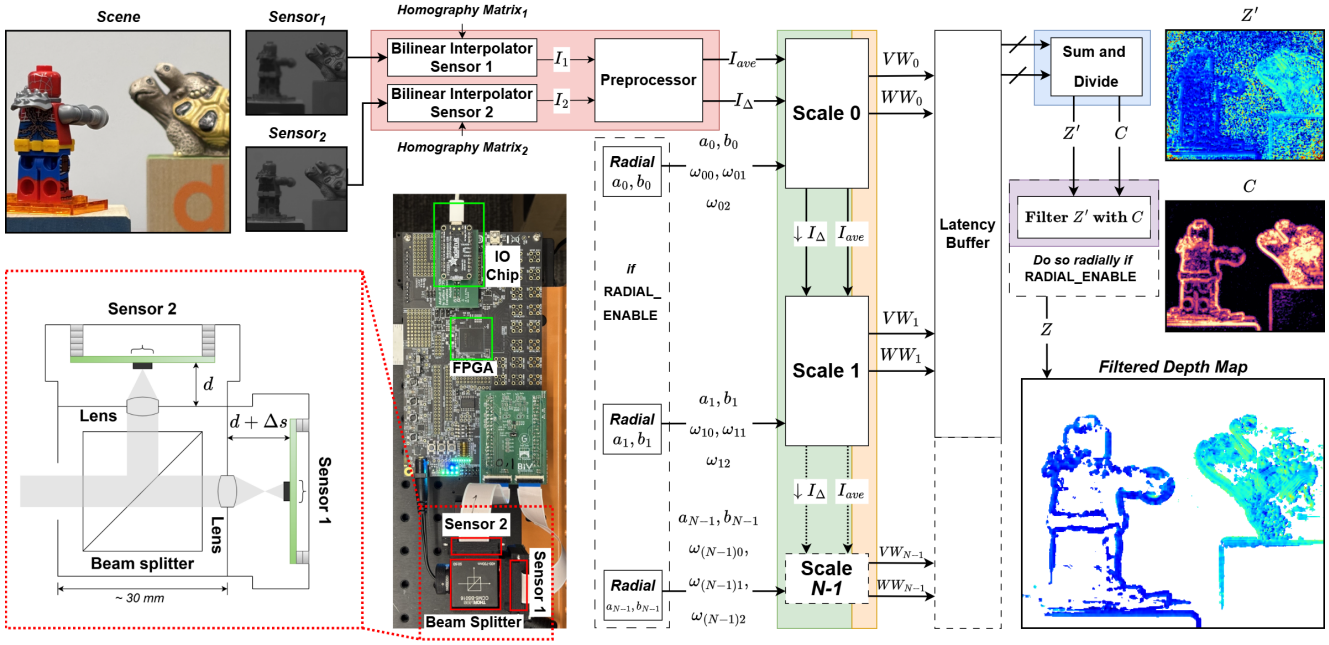

Figure 1. **Method overview.** Our camera uses a beam splitter and a pair of differentially-defocused low power sensors to observe the same scene with an offset depth of field. Our algorithm processes these images to produce depth maps thresholded by confidence.

rithm to adapt the problem to a systolic array, allowing for an efficient solution [55]. While most of these works evaluate their implementations entirely in simulation on existing datasets, Mattoccia *et al*. demonstrated the real-world efficacy of these techniques by implementing a working, compact stereo depth camera [45]. Example depth maps from their device appear dense and accurate, but no quantitative metrics are reported so the working range cannot be evaluated.

2.3. Depth from Defocus

Depth from defocus (DfD) estimates scene depth by analyzing image defocus [53]. Traditional DfD approaches rely on computationally intensive convolution and deconvolution operations, resulting in high computational complexity [38, 67, 70]. Raj & Staunton [30] implement an efficient rational-filter-based DfD algorithm [56] on an FPGA, reporting high real-world depth accuracy (0.5% error) but over a very small working range (77-80 cm from the camera). Their system power is reported as 2 W, but this does not include sensor power or account for sequential rather than snapshot capture.

Our method falls into the DfDD class of algorithms [2–4, 19, 20, 41, 42]. Although DfDD has been shown to require dramatically fewer FLOPs than rational filter methods [41], its potential for low-power hardware implementation remains unexplored. To date, the lowest reported power consumption for a real-time DfDD sensor is still approximately 5 W [42].

3. Method

Our camera uses a beam splitter and two low-power sensors to create a differential defocus signal that is processed into a depth map on a low-power FPGA. Similar to the optical design of Focal Split [42], one sensor is placed slightly farther from the beam splitter (see Fig. 1), resulting in an offset depth of field. Typical of differential defocus methods [1–4, 19, 20, 42], comparisons I_Δ of brightness at corresponding pixels can be compared to spatial variations $\nabla^2 I$ to cheaply reveal scene depth Z at each pixel. Here, we simultaneously collect differentially defocused images I_1 and I_2, which are subtracted for I_Δ and averaged before the spatial derivatives in $\nabla^2 I$. Then,

$$\begin{aligned} Z(x,y) &= V(x,y)/W(x,y), \\ V(x,y) &= a\nabla^2 \tilde{I}(x,y), \\ W(x,y) &= bV(x,y) - \tilde{I}_\Delta(x,y), \end{aligned} \tag{1}$$

where a and b are calibrated camera parameters discussed in Sec. 3.6 and $\tilde{I}$ can refer to the original image, its spatial derivatives, or downsampled versions of the same [19]. In natural scenes, depth cues are sparse but their presence and quality are visible in image data, allowing us to compute confidence for each depth estimate. In practice, several $\tilde{I}$ variations are used to densify and robustify the output depth

map, with a joint depth estimate thresholded by a joint confidence metric for the final depth map.

We introduce novel improvements to the camera and algorithm design before describing an efficient hardware implementation that enables 1.8× power savings over SOTA for the computation alone and 3.3× power savings over SOTA when measuring power consumed by the entire system operating on real hardware.

3.1. Camera design

Our overall camera design follows Focal Split [42], with a few changes to accommodate the smaller size of the low-power sensors (HM0360 [23], which have a 12× power reduction vs. sensors in [42]). As shown in the system photograph in Fig. 1, a 3D-printed enclosure connects each low power sensor to a lens on the beam splitter. In contrast to [42], each sensor has its own lens, resulting in a FoV of 9.4° horizontal × 7.9° vertical (vs. 5.8° × 4.3° for [41]). A 10 mm achromatic lens was selected to strike a balance between a decent FoV and tolerable optical aberration. We provide a detailed parts list and DIY guide for assembling and calibrating our prototype in the supplement.

3.2. Algorithm

Our algorithm, outlined in Fig. 1 and Algorithm 1, processes the images in several stages. First, a preprocessor applies a subpixel-accurate homography to correct for slight misalignments expected in camera assembly, followed by optional denoising by spatial filtering. The images are summed for the average image I_{ave} and differenced for the change image I_Δ. For each of two image scales, V and W of Eq. (1) are computed for $\tilde{I} \in \{I, I_x, I_y\}$. Each $\tilde{I}$ contributes a separate hypothesis (V_i, W_i) in parallel, which are partially summed before the latency buffer aligns the pixel streams across scales. The joint depth and confidence computation solves for depth and confidence from all 6 estimates at each pixel with a novel method described below. Finally, a novel spatially-varying confidence threshold is applied, producing the output depth map.

3.3. Joint Depth Estimation

Previous DfDD methods [19, 20] have combined multiple per-pixel depth estimates $\{Z_i(x,y)\}$ with softmax weighted by a per-pixel, per-estimate, dynamically computed confidence $\{C_i(x,y)\}$:

$$Z(x,y) = \frac{\sum_i e^{C_i(x,y)} Z_i(x,y)}{\sum_i e^{C_i(x,y)}}. \tag{2}$$

While successful in densifying and robustifying output depth, this expression is impossible to compute under our resource constraints. Efficient softmax hardware implementation is an active area of research; without cutting-edge optimizations, the softmax operation would consume more FPGA resources than the entirety of some configurations of our system [15, 27]. Instead, we compute a joint depth estimate from 6 estimates i (based on 2 scales × 3 derivative orders) using calibrated weights ω_i as

$$Z(x,y) = \frac{\sum_i \omega_i V_i(x,y) W_i(x,y)}{\sum_i \omega_i W_i(x,y)^2}. \tag{3}$$

Note that Eq. (3) requires only a single division, which is important because division costs as much as 10 additions in our setting. Improvements to the working range due to this joint estimation can be seen by comparing to the green curve in Fig. 3a and the small blue icons in Fig. 4.

3.4. Confidence Estimation

Because triangulation cues are sparse in natural scenes, a per-pixel confidence score is needed to exclude non-informative image regions where Eq. (1) will provide incorrect output. Previous DfDD methods have used more

Algorithm 1 The SpiderCam Algorithm. This DfDD algorithm achieves SOTA power efficiency with memory locality, effective use of efficient kernels, and robustness to the elevated noise levels of low-power sensors.

Input: images I_1, I_2, calibrated parameters $(\{a_0, ..., a_{N-1}\}, \{b_0, ..., b_{N-1}\}, \{\omega_0, ..., \omega_{(3N-1)}\}, C_{thresh}, Z_{min}, Z_{max})$
Output: depth map $Z(x,y)$
$I_{ave} \leftarrow \texttt{denoise}(I_1 + \texttt{homography}(I_2))/2$
$I_\Delta \leftarrow \texttt{denoise}(I_1 - \texttt{homography}(I_2))/2$
$V, W, VW, WW \leftarrow [\ ]$
$VW_+, WW_+ \leftarrow [\ ]$
for $scale$ in $\{0, 1, ..., N-1\}$ **do**
  $I_{lap} \leftarrow \texttt{laplacian}(I_{ave})$
  for d in $\{none, \partial_x, \partial_y\}$ **do**
    V.push$(a_{scale} \times \texttt{deriv}(I_{lap}, d))$
    W.push$(b_{scale} \times V[-1] - \texttt{deriv}(I_\Delta, d))$
  end for
  for i in $\{scale \times 3, scale \times 3 + 2\}$ **do**
    VW.push$(\omega_i \times V[i] \times W[i])$
    WW.push$(\omega_i \times W[i] \times W[i])$
    $VW[i] \leftarrow \texttt{upsample}(VW[i], scale)$
    $WW[i] \leftarrow \texttt{upsample}(WW[i], scale)$
  end for
  $I_{ave} \leftarrow \texttt{downsample}(I_{ave}, 1)$
  $I_\Delta \leftarrow \texttt{downsample}(I_\Delta, 1)$
  VW_+.push$(\texttt{sum}(VW[scale \times 3 : scale \times 3 + 2]))$
  WW_+.push$(\texttt{sum}(WW[scale \times 3 : scale \times 3 + 2]))$
end for
$C \leftarrow \texttt{sum}(VW_+)$
$Z \leftarrow \texttt{divide}(C, \texttt{sum}(WW_+))$
$Z[C < C_{thresh}$ or not $Z_{min} < Z < Z_{max}] \leftarrow \texttt{null}$

expensive [19, 20] or less accurate [41, 42] confidence metrics. We show that a cheap and effective confidence combining across estimates i at each pixel is:

$$C_i(x,y) = V_i(x,y)W_i(x,y), \tag{4}$$

$$C(x,y) = \sum_i \omega_i C_i(x,y). \tag{5}$$

This simple sum enables us to robustify and densify our measurements across estimates as in [19, 20] without the added expense of their more complex per-pixel estimates or the softmax-based combination previously used. In comparison to the estimator used in [41, 42], we report an increase in working range of 5cm (See Fig. S6). We note that as the numerator of our depth estimation, calculating C requires no additional computation. Low confidence regions will occur when texture is unavailable (so $V \propto \nabla^2 I$ is close to zero) and unstable divisions (when W and therefore the denominator W^2 is close to zero).

3.5. Calibration

Our depth computation requires a handful of calibrated parameters, specifically a and b from Eq. (1), $\{\omega_i\}$ in Eq. (3) and Eq. (5), and thresholds $C_{thresh}, Z_{min}, Z_{max}$ applied to the output. Each depth estimate uses a and b, which physically correspond to

$$a = -A^2, \quad b = (\rho - 1/s)\,, \tag{6}$$

for aperture diameter A, lens optical power ρ, and lens-to-sensor distance s [42]. While these parameters could be derived directly camera measurements, we instead optimize them for depth performance. Calibration of $\{\omega_i\}$ is also data-driven. We collect a real-world calibration dataset by placing a plane with printed texture at 56 known depths, evenly spaced from 0.24m to 1.36m, and optimize the parameters using mean absolute depth error as the loss function. To improve accuracy, we only compute the loss on the 10% most confident pixels in each radial region for each image pair to prevent overfitting to noise. Finally, we establish confidence thresholds for each radial zone to maximize the working range, see Sec. S10 for details.

3.6. Spatial Variation

The optics available for the compact size of our low-power sensors create noticeable spatial variation in defocus due to Petzval field curvature [22] and other nonidealities. Because of this, performance suffers if we hold our a, b, and thresholds to constant values across the image sensor. We improve performance by allowing calibrated parameters to vary for 16 separate radial zones of equal width. Since the pixels arrive in order in the pixel stream, we are able to track the radial locations of each pixel without having to actually buffer the (x,y) information alongside each pixel value. By comparing a streaming value to thresholds on squared radial distances (to avoid the expensive square root operation), we can dynamically update the values of $(\{a_0, ..., a_{N-1}\}, \{b_0, ..., b_{N-1}\}, C_{thresh}, Z_{min}, Z_{max})$, see in detail Sec. S1.5. Note that ω_i values do not vary spatially. Accounting for spatial variation results in a larger working range (Fig. 3a) and better calibrated depth for each radial region (Fig. S4) with only a small increase in core power (Fig. 4).

3.7. Efficient Hardware Implementation

FPGA size and resource utilization strongly affects system power consumption - larger FPGAs have higher static power dissipation. In order to make our system as efficient as possible, we designed our hardware to fit inside the Lattice ECP5 LFE5U-85F FPGA. By using a smaller FPGA, the benefits of our efficient algorithm implementation are realized in power savings.

We also had to restrict our memory use to achieve low-power operation. Low-power FPGAs have extremely limited memory budgets. Due to the high energy cost of data movement, extending their memory by using off-chip DRAM incurs a prohibitively high energy cost [17]. The LFE5U-85F does not have enough on-chip memory to hold both frames at once. Adding enough storage to buffer both frames without moving to a larger, more power-hungry FPGA would require the use of external DRAM, raising power consumption and incurring depth map latency. This means that we must process our depth maps with a data streaming model, where the data locality of the DfDD algorithm enables significant power savings. Additionally, we developed a hybrid fixed and floating point architecture that supports the high dynamic range of natural scenes while minimizing per-pixel memory cost.

3.7.1. Multiscale compute within fixed memory budget

Our robust depth algorithm requires upsampling in two of its computations: estimating $\nabla^2 \tilde{I}$ with a Laplacian pyramid, and merging downsampled estimates into our multiscale joint estimation.

Downsampling a streaming image is straightforward, but naive upsampling requires buffering a prohibitive amount of the image. We develop a specialized streaming upsampler using zero-interleaved kernels that reach a larger effective receptive field without increasing arithmetic operations. This both decreases the number of lines buffered and fixes the memory use to be independent of image height. See Sec. S4 for details.

3.7.2. Efficient Kernels

We minimize FPGA power and memory costs by using kernels that are small, have integer coefficients that are powers of two, and are linearly separable. This, respectively, minimizes the number of lines buffered, converts multiplication

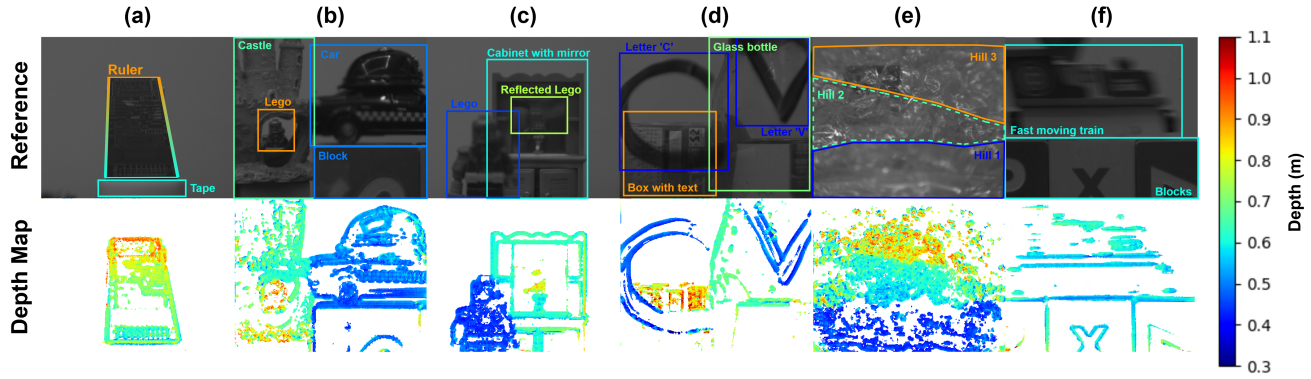

Figure 2. **Captured images and depth maps computed on-device.** Note the handling of continuous depths (vs. disparity levels), reflective textures, and transparent objects that active sensing would struggle with, as well as objects in motion due to snapshot capture.

to bit-shifting, and reduces convolution costs by a factor of kernel width. We use box, Gaussian, and derivative kernels in the preprocessing and per-scale computations, with our largest kernel being a 5-tap Burt-Adelson Gaussian filter [9]. See Sec. S2.1 for details.

3.7.3. Hybrid Fixed Point with Efficient Floating Point

Fixed point arithmetic is cheap in hardware, but limits the dynamic range of values that can be represented accurately. Most vision applications require floating point, which is significantly more expensive. We reduce this cost by identifying preprocessing steps that can be computed in fixed point without loss of accuracy, then switching to an efficient floating point implementation that maximizes accuracy within a constrained memory footprint.

Specifically, the homography and preprocessing stages are implemented in fixed point. The homography only requires a small number of lines to be buffered since the affine transformation is small. In the preprocessing step, our carefully selected filters do not require bit trimming, avoiding quantization issues. Due to the high dynamic range of our images, after these stages, the results are converted to floating point before computing I_{ave} and I_{Δ}.

On the floating-point side, we save FPGA resources by dropping subnormal number support and using FP16 instead of FP32. Removing subnormal number support makes multipliers and dividers nearly as inexpensive as fixed-point arithmetic, with only a small additional cost for exponent handling. Our adders still require the variable shifters and priority encoders associated with floating-point arithmetic, but some of the surrounding logic is simplified by removing subnormal handling. The primary reduction in resource usage for our floating-point adders comes from using half-precision (FP16) representation. See Sec. S3 for details.

3.7.4. Data Readout

Our system outputs 8-bit depth maps over a parallel port, similar to how an image sensor outputs its data. To read the results of the depth computation from the camera, we use an FTDI FT232H parallel-to-USB interface chip. Our communication speed is limited to 8MB/s by the design of the ECP5 development board that our prototype system uses, meaning that in-situ our system can output depth maps at a maximum of 32.5 FPS. Alternatively, for data collection and demonstration, we can read out image pairs or one image with its respective depth map at a slower rate. Without these I/O limitations, our core computation could run up to 140 FPS on ECP5 and up to 300 FPS on Kintex-7.

4. Results

Depth maps of real scenes computed on-device are shown in Fig. 2, along with corresponding sensor images. Our computed confidence excludes low-signal regions, returning accurate depth estimates around scene texture. Note that our method can gracefully handle scenes with multiple depths, including continuous (a), discontinuous (b), and reflected (c) depths. Our sparse depth maps gracefully handle overlapping transparent objects (d), our passive method easily handles bubble wrap (e) even though it is transparent, reflective, and non-rigidly deformed, and our snapshot camera returns accurate depth even in the presence of significant motion blur (f). Additional scenes and videos are provided in the supplement.

We characterize the performance of the system using front-parallel textured planes across densely-sampled depths, and define the working range (WR) as the set of depths for which $|Z_{est} - Z_{true}| < 0.1\ Z_{true}$, with our device achieving a working range of 0.45-0.97 m. Accuracy of on-device computation (purple dots) shows good agreement with off-device computation (purple and all other lines) in Fig. 3a. Off-device computation is slightly more accurate because it is performed fully in 32-bit floating point for convenience and fairness in comparing across algorithms.

In comparison to our method (offline) with a working

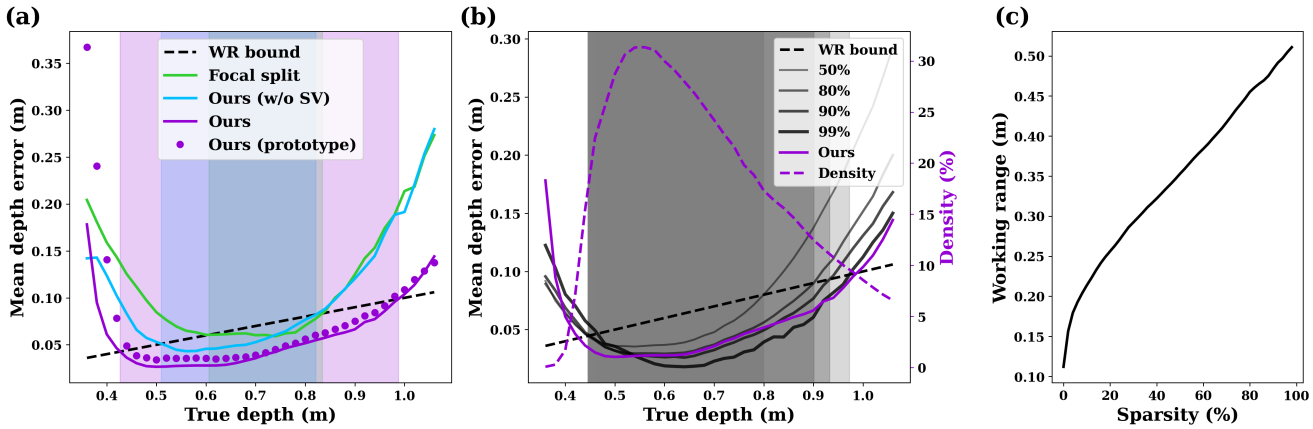

Figure 3. **Quantitative analysis on images from device.** (a) Mean absolute error (MAE) of depth as a function of true depth, with working range bounds. Our method robustly outperforms the Focal Split algorithm [42] in the low power hardware setting, and accounting for spatial variation in our compact optics is vital. Purple dots show errors from depth maps computed on-device and demonstrate good agreement with off-line depth computations (rest of figure), which were performed in 32-bit floating point. (b) The MAE vs. depth curve of our method with dynamic sparsity values set by per-image confidence percentiles (grays) and the MAE (solid purple) and output pixel density (100-sparsity, dashed purple) resulting from the fixed confidence thresholds used in our method. (c) Working range increases as data is sparsified by per-image confidence percentiles.

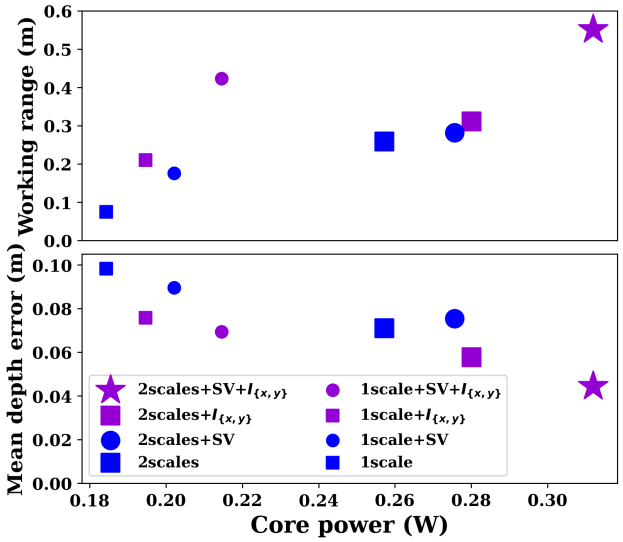

Figure 4. **Power/accuracy trade-offs for alternate algorithms.** We estimate max core power on the ECP5 with manufacturer's tools and evaluate working range and MAE from 0.4m to 1.0m offline on images collected by our hardware prototype. Adding computational components improves accuracy while increasing power cost. Our algorithm is shown with the purple star.

range (pale purple highlight) of 0.43-0.99 m, the Focal Split algorithm [42] (green) has a working range of only 0.60-0.83 m on the challenging low-power camera data. Spatial variation is a major contributor to this improvement, as removing it (blue) drops our working range to 0.51-0.82 m.

Next, to explore the effect of confidence on sparsity and accuracy, Fig. 3b shows the MAE and WR for depth- and scene-varying confidence thresholds set to enforce a fixed sparsity across each depth map (gray levels). Our method, repeated in purple for reference, takes a fixed confidence threshold that naturally loses density (100% - sparsity, dashed purple) at the edge of the WR. Note that fixed thresholds result in different error/density trade-offs than enforced sparsities, highlighting the need for computationally inexpensive, well-calibrated confidence estimation as an area for future research. Fig. 3c shows the size of the working range across enforced sparsities.

4.1. Resource Utilization and Power Consumption

DfDD core power and size. To evaluate the efficiency of the SystemVerilog DfDD implementation itself, we synthesized multiple configurations of the DfDD core depth computation. Even though our physical system is implemented with a Lattice ECP5 FPGA, for the sake of this specific resource usage comparison, we synthesized our design for the more widely-used Xilinx 7-Series FPGA architecture, which is widely compatible across the literature. Resource utilization results can be seen in Fig. 5, details in Sec. S8. Each FPGA has a limited number of lookup tables (LUT), registers (REG), and block RAM (BRAM) with which a circuit design can be implemented. A smaller design consumes less resources and can fit on smaller FPGAs, allowing for lower power consumption and better system miniaturization. We also took these utilization numbers and used vendor tools to estimate power consumption, comparing our system's power consumption with similar work Tab. 1.

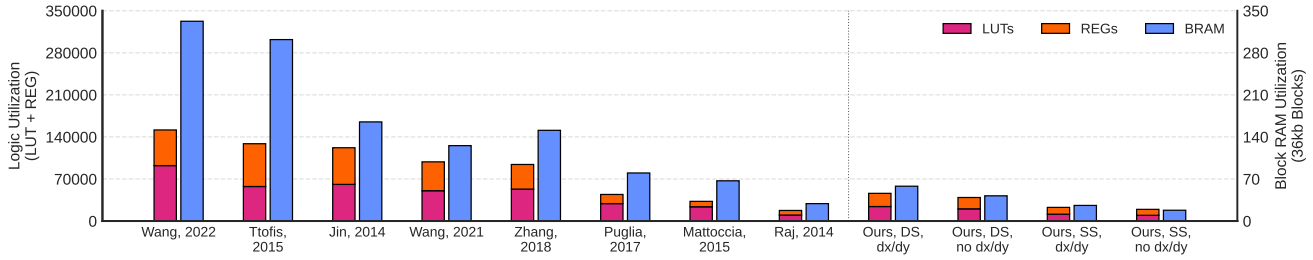

Figure 5. **Resource Usage Across Methods.** We compare the resource utilization of several configurations of our system as well as systems in the literature. "DS" and "SS" denote evaluation at 2 and 1 scales, respectively.

Power consumption of hardware camera. Critical to our work is the construction of an actual hardware camera capable of producing depth images in real-time at over 30 FPS. Our hardware camera consisted of two HM0360 low-power image sensors in a custom optical setup (as described in Sec. 3.1) connected via a custom interposer PCB to a commercially available Lattice ECP5 evaluation board [35] equipped with a LFE5UM5G-85F FPGA [36]. In order to reduce power consumption, some modifications were made to the evaluation board. A summary of these modifications can be found in Sec. S9.1. To measure power consumption, we powered the camera entirely by an Otii Arc power profiler and operated it under normal conditions, recording natural scenes, integrating average power consumption over 5 seconds. Our results are shown across configurations in Tab. S6. Significantly, our full algorithm consumes only 624 mW at 32.5 FPS and 399 mW at 9 FPS, 3.3 - 5× less power than the lowest full system power reported in the literature [55], which runs at 30 FPS.

Total system power measurements in Tab. S6 are higher than core power estimates in Tab. 1 and Fig. S5 because the physical hardware system also includes: two image sensors and image homography and cropping logic, I/O circuitry, and the evaluation board which contains several unused components and introduces inefficiencies in, e.g., voltage converters.

4.2. Accuracy Across Restricted Power Budgets

We analyze ablations of several subcomponents in terms of both depth performance, evaluated by working range and depth MAE over 0.4 m to 1.0 m (computed off-device), and core power consumption (estimated for the ECP5 from manufacturer's tools). Fig. 4 provides an overview of these trade-offs. Adding spatial variation in calibrated parameters improves working range and reduces MAE (see corresponding square-circle pairs). As expected, the accuracy among well-calibrated (i.e. spatially varying) methods goes according to the number of separate estimates used in the depth map: the best has 6 (purple star: 2 scales, 3 derivatives), the next best has 3 (small purple circle: 1 scale, 3 derivatives), the next best has 2 (large blur circle: 2 scales, 1 derivative), and the worst has only 1 (small blue circle: 1 scale, 1 derivative). Adding derivatives is cheaper in terms of power consumption than adding image scales, though we note that the relative cost appears more extreme in estimates of core power than the true relative difference in the overall system, since peripheral power costs dominate and will be constant across methods. Specifically, in comparing our method with the next best option (single scale, multiderivative, shown by the purple circle), we estimate a core power difference of 311 vs. 214 mW for a roughly 32% difference, but when running the hardware prototype at 32.5 FPS we measure the full systems' wall power at 624 vs. 489 mW – only a 20% difference (dropping to an 11% difference at 9 FPS, see Tab. S6). At a high level, this study indicates the potential of our method to provide depth estimates for a range of power constraints through appropriate computational trade-offs.

5. Conclusion

We present SpiderCam, the first sub-Watt FPGA-based passive 3D camera. Featuring snapshot capture with compact off-the-shelf optics, low-power sensors, a robust and efficient new DfDD algorithm, and a SystemVerilog implementation on a low-power FPGA, we demonstrate a working range of 52 cm at 32.5 FPS in real time on real data with a full-system wall power of 624 mW. We have characterized the power/accuracy trade-offs of several variations of our system, and identified improvements in optics and confidence estimation as immediate future directions for system improvement. Significant gains in power consumption could be pursued by moving from an FPGA to an ASIC, which typically results in 5-20× power savings depending on design details [31, 43]. Our sparse and somewhat noisy depth estimates offer exciting possibilities as control signals for embedded/edge applications like wearables and robotics, which will require further research and development. An immediate direction for future work is the integration of our outputs with efficient depth-completion pipelines like [72] and increasing the FoV of our system.

Acknowledgments

This research was partially supported by the National Science Foundation under awards numbers CNS-2430327 and CCF-2431505. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or other supporters. We would also like to thank Junjie Luo and Alan Fu for helpful discussions.

References

[1] Emma Alexander. *A theory of depth from differential defocus*. Harvard University, 2019. 1, 3

[2] Emma Alexander, Qi Guo, Sanjeev Koppal, Steven Gortler, and Todd Zickler. Focal flow: Measuring distance and velocity with defocus and differential motion. In *European conference on computer vision*, pages 667–682. Springer, 2016. 3

[3] Emma Alexander, Qi Guo, Sanjeev Koppal, Steven J Gortler, and Todd Zickler. Focal flow: Velocity and depth from differential defocus through motion. *International Journal of Computer Vision*, 126(10):1062–1083, 2018.

[4] Emma Alexander, Leyla A Kabuli, Oliver S Cossairt, and Laura Waller. Depth from defocus as a special case of the transport of intensity equation. In *2021 IEEE International Conference on Computational Photography (ICCP)*, pages 1–13. IEEE, 2021. 3

[5] Jason Helge Anderson and Farid N Najm. Power estimation techniques for fpgas. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(10):1015–1027, 2004. 6

[6] Stephen T Barnard and Martin A Fischler. Computational stereo. *ACM Computing Surveys (CSUR)*, 14(4):553–572, 1982. 2

[7] Behnam Behroozpour, Phillip AM Sandborn, Ming C Wu, and Bernhard E Boser. Lidar system architectures and circuits. *IEEE Communications Magazine*, 55(10):135–142, 2017. 1

[8] Amlaan Bhoi. Monocular depth estimation: A survey. *arXiv preprint arXiv:1901.09402*, 2019. 2

[9] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987. 6, 2

[10] Vijay Degalahal and Tim Tuan. Methodology for high level estimation of fpga power consumption. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 657–660, 2005. 6

[11] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013. 1

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1

[14] Jason Geng. Structured-light 3d surface imaging: a tutorial. *Advances in optics and photonics*, 3(2):128–160, 2011. 1

[15] Xue Geng, Jie Lin, Bin Zhao, Anmin Kong, Mohamed M Sabry Aly, and Vijay Chandrasekhar. Hardware-aware softmax approximation for deep neural networks. In *Asian Conference on Computer Vision*, pages 107–122. Springer, 2018. 4

[16] Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W Mahoney, and Kurt Keutzer. Ai and memory wall. *IEEE Micro*, 44(3):33–39, 2024. 2

[17] Saugata Ghose, Abdullah Giray Yaglikçi, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X Liu, Hasan Hassan, Kevin K Chang, Niladrish Chatterjee, Aditya Agrawal, et al. What your dram power models are not telling you: Lessons from a detailed experimental study. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(3):1–41, 2018. 5

[18] Silvio Giancola, Matteo Valenti, and Remo Sala. *A survey on 3D cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies*. Springer, 2018. 1, 2

[19] Qi Guo, Emma Alexander, and Todd Zickler. Focal track: Depth and accommodation with oscillating lens deformation. In *Proceedings of the IEEE international conference on computer vision*, pages 966–974, 2017. 3, 4, 5

[20] Qi Guo, Zhujun Shi, Yao-Wei Huang, Emma Alexander, Cheng-Wei Qiu, Federico Capasso, and Todd Zickler. Compact single-shot metalens depth sensors inspired by eyes of jumping spiders. *Proceedings of the National Academy of Sciences*, 116(46):22959–22965, 2019. 3, 4, 5

[21] Rostam Affendi Hamzah and Haidi Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016(1):8742920, 2016. 2

[22] Eugene Hecht. *Optics*. 2017. 5, 10

[23] *Datasheet 1/6" 640 x 480 VGA 60FPS CMOS Image Sensor*. Himax Technologies, 2020. 4

[24] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. 2

[25] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007. 1

[26] Radu Horaud, Miles Hansard, Georgios Evangelidis, and Clément Ménier. An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine vision and applications*, 27(7):1005–1020, 2016. 1

[27] Ruofei Hu, Binren Tian, Shouyi Yin, and Shaojun Wei. Efficient hardware architecture of softmax layer in deep neural network. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–5. IEEE, 2018. 4

[28] *Intel RealSense D400 Series Product Family Datasheet*. Intel, 2019. 2

[29] Minxi Jin and Tsutomu Maruyama. Fast and accurate stereo vision system on fpga. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 7(1):1–24, 2014. 2, 8

[30] Alex Noel Joseph Raj and Richard C Staunton. Video-rate calculation of depth from defocus on a fpga. *Journal of Real-Time Image Processing*, 14(2):469–480, 2014. 2, 3, 8

[31] Ian Kuon and Jonathan Rose. Measuring the gap between fpgas and asics. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, pages 21–30, 2006. 8

[32] Hamid Laga, Laurent Valentin Jospin, Farid Boussaid, and Mohammed Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE transactions on pattern analysis and machine intelligence*, 44(4):1738–1764, 2020. 2

[33] MF Land. Structure of the retinae of the principal eyes of jumping spiders (salticidae: Dendryphantinae) in relation to visual optics. *Journal of experimental biology*, 51(2):443–470, 1969. 1

[34] *Power Consumption and Management for ECP5 and ECP5-5G Devices*. Lattice. 6

[35] *ECP5 Evaluation Board User Guide*. Lattice Semiconductor, 2018. 8

[36] *ECP5 and ECP5-5G Family Data Sheet*. Lattice Semiconductor, 2025. 8

[37] Sanghoon Lee, Dongkyu Lee, Pyung Choi, and Daejin Park. Accuracy–power controllable lidar sensor system with 3d object recognition for autonomous vehicle. *Sensors*, 20(19): 5706, 2020. 2

[38] Anat Levin, Rob Fergus, Frédo Durand, and William T Freeman. Image and depth from a conventional camera with a coded aperture. *ACM transactions on graphics (TOG)*, 26 (3):70–es, 2007. 3

[39] MultiMedia LLC. Product info - geforce gtx titan x, 2025. 2

[40] Zhimin Lu, Jue Wang, Zhiwei Li, Song Chen, and Feng Wu. A resource-efficient pipelined architecture for real-time semi-global stereo matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(2):660–673, 2021. 2, 8

[41] Junjie Luo, Yuxuan Liu, Emma Alexander, and Qi Guo. Depth from coupled optical differentiation: J. luo et al. *International Journal of Computer Vision*, pages 1–18, 2025. 1, 3, 4, 5, 10

[42] Junjie Luo, John Mamish, Alan Fu, Thomas Concannon, Josiah Hester, Emma Alexander, and Qi Guo. Focal split: Untethered snapshot depth from differential defocus. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26965–26974, 2025. 2, 3, 4, 5, 7, 9, 10, 11

[43] John Mamish, Rawan Alharbi, Sougata Sen, Shashank Holla, Panchami Kamath, Yaman Sangar, Nabil Alshurafa, and Josiah Hester. Nir-sighted: A programmable streaming architecture for low-energy human-centric vision applications. *ACM Transactions on Embedded Computing Systems*, 23(6): 1–26, 2024. 8

[44] Armin Masoumian, Hatem A Rashwan, Julián Cristiano, M Salman Asif, and Domenec Puig. Monocular depth estimation using deep learning: A review. *Sensors*, 22(14): 5353, 2022. 2

[45] Stefano Mattoccia and Matteo Poggi. A passive rgbd sensor for accurate and real-time depth sensing self-contained into an fpga. In *Proceedings of the 9th International Conference on Distributed Smart Cameras*, pages 146–151, 2015. 1, 2, 3, 7, 8, 11

[46] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[47] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021. 2

[48] Parsa Mirdehghan, Wenzheng Chen, and Kiriakos N Kutulakos. Optimal structured light a la carte. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6248–6257, 2018. 1

[49] Takashi Nagata, Mitsumasa Koyanagi, Hisao Tsukamoto, Shinjiro Saeki, Kunio Isono, Yoshinori Shichida, Fumio Tokunaga, Michiyo Kinoshita, Kentaro Arikawa, and Akihisa Terakita. Depth perception from image defocus in a jumping spider. *Science*, 335(6067):469–471, 2012. 1

[50] Dat Ngo, Hyun-Cheol Park, and Bongsoon Kang. Edge intelligence: A review of deep neural network inference in resource-limited environments. *Electronics*, 14(12):2495, 2025. 2

[51] Matthew O'Toole, John Mather, and Kiriakos N Kutulakos. 3d shape and indirect appearance by structured light transport. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3246–3253, 2014. 1

[52] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion*. *Acta Numerica*, 26:305–364, 2017. 2

[53] Alex Paul Pentland. A new sense for depth of field. *IEEE transactions on pattern analysis and machine intelligence*, (4):523–531, 1987. 3

[54] Matteo Poggi, Fabio Tosi, Konstantinos Batsos, Philippos Mordohai, and Stefano Mattoccia. On the synergies between machine learning and binocular stereo for depth estimation from images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5314–5334, 2021. 2

[55] Luca Puglia, Mario Vigliar, and Giancarlo Raiconi. Real-time low-power fpga architecture for stereo vision. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64 (11):1307–1311, 2017. 1, 2, 3, 8, 7, 11

[56] Alex Noel Joseph Raj and Richard C Staunton. Rational filter design for depth from defocus. *Pattern Recognition*, 45(1): 198–207, 2012. 3

[57] Thinal Raj, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain. A survey on lidar scanning mechanisms. *Electronics*, 9(5):741, 2020. 2

[58] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018. 2

[59] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007. 1

[60] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[61] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, pages I–I. IEEE, 2003.

[62] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014. 1

[63] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 2

[64] Md Maruf Hossain Shuvo, Syed Kamrul Islam, Jianlin Cheng, and Bashir I Morshed. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. *Proceedings of the IEEE*, 111(1):42–91, 2022. 2

[65] Christos Ttofis, Christos Kyrkou, and Theocharis Theocharides. A low-cost real-time embedded stereo vision system for accurate disparity estimation based on guided image filtering. *IEEE Transactions on Computers*, 65(9):2678–2693, 2015. 1, 2, 8

[66] Hongyu Wang, Wei Zhou, Xiangyu Zhang, and Xin Lou. A block patchmatch-based energy-resource efficient stereo matching processor on fpga. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(7):2893–2905, 2022. 1, 2, 8

[67] Masahiro Watanabe and Shree K Nayar. Rational filters for passive depth from defocus. *International Journal of Computer Vision*, 27(3):203–225, 1998. 3

[68] *Xilinx Power Estimator User Guide*. Xilinx. 6

[69] Xuchong Zhang, Hongbin Sun, Shiqiang Chen, Lin Song, and Nanning Zheng. Nipm-swmf: Toward efficient fpga design for high-definition large-disparity stereo matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(5):1530–1543, 2018. 2, 8

[70] Changyin Zhou, Stephen Lin, and Shree K Nayar. Coded aperture pairs for depth from defocus and defocus deblurring. *International journal of computer vision*, 93(1):53–72, 2011. 3

[71] Kun Zhou, Xiangxi Meng, and Bo Cheng. Review of stereo matching algorithms based on deep learning. *Computational intelligence and neuroscience*, 2020(1):8562323, 2020. 2

[72] Yiming Zuo, Willow Yang, Zeyu Ma, and Jia Deng. Omnidc: Highly robust depth completion with multiresolution depth integration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9287–9297, 2025. 8