Stereo Depth Estimation

Bryan Hau-Ping Chiang
bhchiang@stanford.edu

Natasha Saki Ong
natashao@stanford.edu

Abstract

Mixed reality systems rely heavily on depth information to provide compelling experiences. At the same time, mixed reality systems experience several constraints including power consumption, latency and form factor, making most state-of-the-art stereo models which rely on costly 3D convolutions difficult to deploy. In this work, we implement an end-to-end-differentiable model that avoids such computationally costly convolutions and compare our model's performance against traditional stereo block matching methods. By implementing a learned stereo matching network, we achieve more robust depth estimation with an average end-point error (EPE) of 1.68 pixels and a 1-pixel error of 38% on our validation set.

Our code is available at https://github.com/bryanhpchiang/sde.

1. Introduction

1.1. Motivation

Depth estimation is a fundamental computer vision task and is integral in applications such as scene understanding, 3D reconstruction, and autonomy. In emerging spatial computing systems such as optical see-through augmented reality (OST-AR) headsets, real-time depth is critical for creating compelling experiences. On one end, depth provides the foundation for metric-semantic reconstruction and dynamic scene graphs, enabling the development of intelligent, context-aware applications [18]. On the other end, compositing hybrid environments in which real and virtual objects are synthesized in real-time requires depth to properly render key focal cues such as occlusion and complex view-dependent effects [1]. The challenge in estimating depth for such mixed reality devices thereby lies in the significant power consumption, latency, and form factor constraints imposed by their wearable and real-time nature.

1.2. Problem

Active methods such as structured light stereo and LIDAR offer high-resolution depth maps, but suffer from limited range, spatial and temporal resolution tradeoff, susceptibility to weather conditions (i.e. broad daylight) and high costs. Consequentially, we focus on passive depth estimation from visual input, specifically stereo image pairs. Depth from stereo is cost-effective and generalizes to wider range of environments at farther distances compared to LIDAR in embedded environments. Mixed reality systems also perform a variety of tasks such as real-time hand tracking, visual-inertial odometry (VIO), and object recognition for which vision is highly useful.

In stereo vision, depth estimation reduces to the dense stereo correspondence problem. Given a rectified stereo pair of images I^L and I^R, we aim to compute the disparity d for each pixel in the reference image. Disparity is defined as the horizontal displacement between corresponding pixels across the pair.

In recent years, traditional stereo matching algorithms have been superseded by modern learning-based approaches that leverage deep neural networks to achieve high accuracies. Modern methods formulate depth estimation as a supervised learning task, replacing traditional steps of the pipeline with differentiable modules such as convolutional neural networks (CNNs) for optimization in an end-to-end manner.

However, current learning-based methods suffer from two key issues:

- Learned models fail to generalize across different camera hardware setups and are thus difficult to deploy for real-world use.
- Cost volume aggregation is typically performed via expensive 3D convolutions, which significantly increase inference times and are not typically supported by embedded hardware accelerators and microcontrollers.

In this work, we implement an stereo depth estimation model designed for actual stereo camera hardware.

2. Related Work

2.1. Monocular Depth Estimation

Monocular-based estimation algorithms infer depth from a single image. Recently, deep learning approaches have

been applied to the monocular depth estimation problem and has seen rapid advancements by exploring image-level information from the neural networks [7]. However, due to the ill-posed nature of the problem and the scarcity of high quality datasets, the quality is generally limited [8].

While monocular depth estimation is a promising method fulfilling hardware and computation constraints, monocular methods suffer from scale ambiguity and are unlikely to achieve sufficient accuracy in unfamiliar scenes. Since metric reconstruction and recovery of details are important aspects in mixed reality systems, stereo matching methods, which estimate depth from stereo pairs using triangulation, may be a more suitable alternative [22].

2.2. Stereo Matching Networks

Current state-of-the-art stereo matching networks achieve high accuracies by relying on direct feature concatenation to form 4D cost volumes suitable for 3D convolutions [13, 15]. In contrast, 2D convolution based methods use feature correlation to form 3D cost volumes, trading off performance for lower computational complexity (up to two orders of magnitude according to [21]). In practice, 3D convolutions are decomposed into 2D and 1D convolutions via depth-wise separable convolutions. In our work, we experiment with 2D deformable convolutions to enable adaptive aggregation by sampling a fixed amount of points over learned positions [6].

3. Methods

3.1. Hardware

Our target hardware is the Intel RealSense T261, an off-the-shelf tracking module consisting of two global shutter fisheye cameras and an inertial measurement unit (IMU). Localization is performed through visual and inertial sensor fusion, with 6 DoF pose estimates at 200 Hz. Its compact form factor, low power consumption, and built-in odometry make it suitable for use in mixed reality systems. The device has a fisheye field of view (FoV) of $163 \pm 5°$, outputting grayscale images at 30 frames per second (fps).

Since our FoV is less than $180°$, fisheye distortion will be handled by applying the Kannala-Brandt model on top of our standard pinhole optics.

4. Architecture

Instead of designing every step of traditional stereo algorithms carefully by hand, we aim to create a model that learns an end-to-end mapping, visualized in Fig. 1. Our model contains differentiable layers representing the major components in traditional stereo matching pipelines [19]:

- matching cost computation
- cost aggregation
- disparity computation
- disparity refinement

4.1. Feature Extraction

In order to compute stereo matching cost, our model first learns the unary feature extraction using shared weights. The left and right rectified stereo images, I_l and I_r respectively, are processed by a feature extractor that follows that proposed in GC-Net [14], an architecture for disparity regression using end-to-end learning of geometry and context.

Features are learned through a number of 2D convolutions, where each convolution is followed by a batch normalization layer and a rectified linear (ReLU) non-linearity. We first apply a 5x5 convolution with 32 features and stride $s = 2$ to reduce the computational demand. Then, we apply 8 basic residual blocks [9], following the residual blocks outlined in PSM-Net [5] (two 3x3 convolutions in series). Finally, a 3x3 convolution without batch normalization or ReLU is applied for an output dimension of $(H/2, W/2, 32)$.

4.2. Cost Volume Computation

We use the features from our feature extractor to construct a cost volume, which largely follows the formulation in DispNetC [16]. The cost volume stores matching costs for a pixel with its corresponding pixels at the next frame [12] along varying disparities. In particular, to increase running speed, we *correlate* left and right features instead of concatenating them:

$$\mathbf{C}(d, h, w) = \frac{1}{N} \langle \mathbf{F}_l(h, w), \mathbf{F}_r(h, w - d) \rangle$$

where $\langle \cdot, \cdot \rangle$ represents the the inner product of two feature vectors, N is the channel number of extracted features, $\mathbf{C}(d, h, w)$ is the matching cost at location (h, w) for disparity candidate d, and $\mathbf{F}_l, \mathbf{F}_r$ are the left and right features, respectively.

4.3. Cost Aggregation

To eliminate the use of 3D convolutions, we adopt the recently proposed adaptive aggregation module [21].

Intra-scale aggregation relies on adaptively sampling neighboring points. Given $\mathbf{C} \in \mathbb{R}^{D \times H \times W}$, where D is the maximum disparity and H and W are the height and width respectively, we define the aggregated cost $\tilde{\mathbf{C}}$ for disparity candidate d at pixel $\mathbf{p} = (x, y)$.

$$\tilde{\mathbf{C}}(d, \mathbf{p}) = \sum_{k=1}^{K^2} w_k \cdot \mathbf{C}(d, \mathbf{p} + \mathbf{p}_k + \Delta \mathbf{p}_k) \cdot m_k \quad (1)$$

K^2 is the number of sampling points, w_k is the k-th aggregation weight, $\mathbf{p}_k$ is the pixel offset to reach pixel $\mathbf{q}$ from

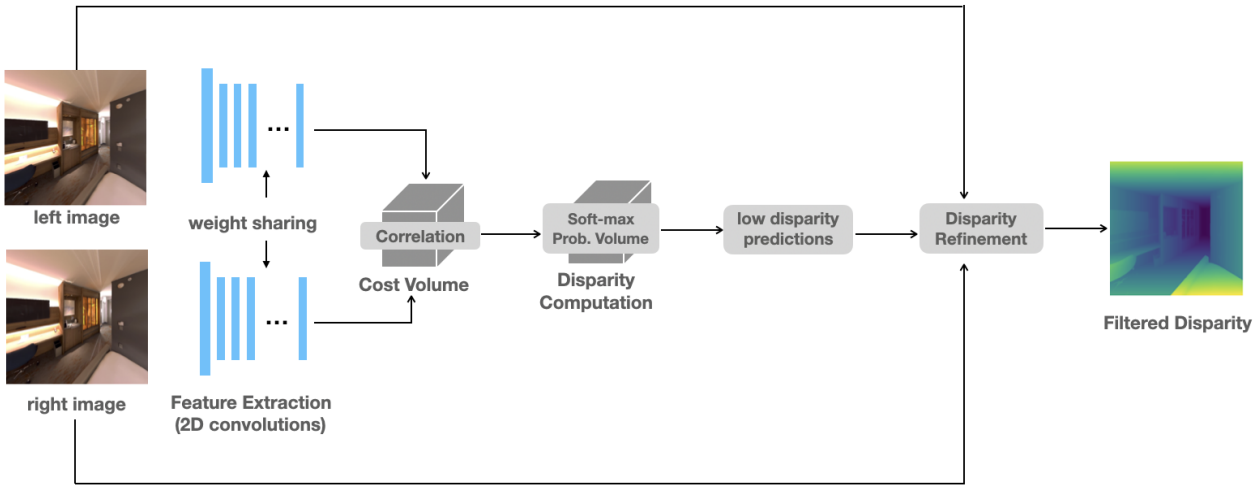

Figure 1. Our Network Architecture Pipeline

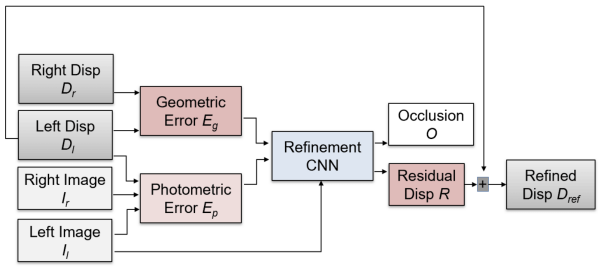

Figure 2. StereoDRNet Refinement Architecture

$\mathbf{p}$ as in traditional window-based aggregation, $\Delta\mathbf{p}_k$ is our new learned offset, and m_k are learned content-adaptive weights. We implement intra-scale cost aggregation as a deformable convolution, in which $\{w_k\}_{k=1}^{K^2}$ are the convolution weights, $\Delta\mathbf{p}_k$ are obtained via a separate convolution over $\mathbf{C}$. Due to memory constraints, we set $m_k = 1$ everywhere.

4.4. Disparity Regression

Following [13], we use the differentiable *soft argmin* operator to regress our final disparity value $\hat{d}$ from a probability volume of all candidates.

$$\tilde{d} = \sum_{k=0}^{D_{\max}} d \cdot \sigma(c_d) \tag{2}$$

d is the disparity candidate, σ is the softmax operator, and c_d is the cost of disparity candidate d.

4.5. Disparity Refinement

For the final step of the pipeline, we perform disparity refinement as proposed in StereoDRNet [4] illustrated by Fig. 2 which uses the geometric error E_g, photometric error E_p and unrefined disparity to produce a refined disparity.

To evaluate the geometric consistency error map, we first warp the right image to left view using a warp and evaluate the image reconstruction error map E_p for the left image:

$$E_p = |I_l - W(I_r, D_r)|$$

where I_l, I_r are the left, right images, respectively, W is the warp that is applied, and D_l, D_r are the predicted disparities.

Then, warping D_r to the left view and using the left disparity D_l, we can get the geometric consistency error map E_g:

$$E_g = |D_l - W(D_r, D_l)|.$$

We filter the concatenated left image and reconstruction error and the left disparity and geometric error map using convolutions followed by batch normalization and relu. We concatenate these results then apply atrous convolutions [17] using respective dilations of $[1, 2, 4, 8, 1, 1]$. Finally, a single 3×3 convolution is used to obtain the refined disparity.

5. Results

5.1. Camera Properties

Using the Intel RealSense SDK, we extracted the intrinsics K_l and K_r, rotation R and translation T from the left to right camera.

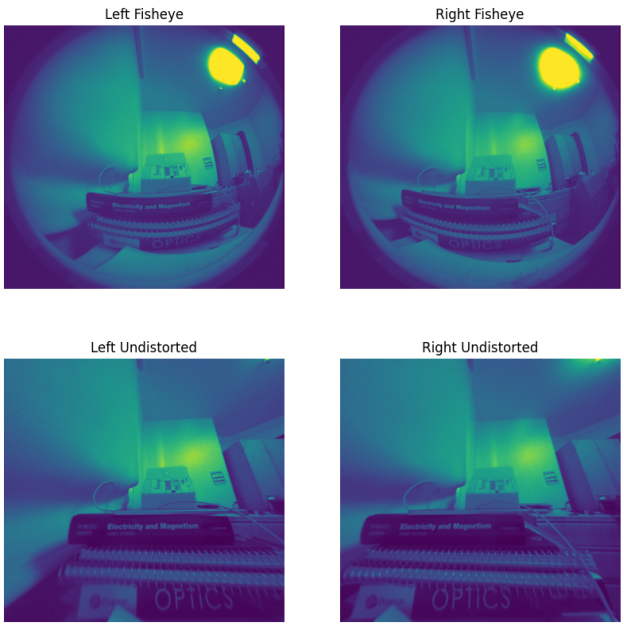

Figure 3. Rectification and undistortion results for both left and right fisheye cameras.

$$K_l = \begin{pmatrix} 287.96 & 0 & 434.93 \\ 0 & 286.32 & 395.52 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$K_r = \begin{pmatrix} 287.90 & 0 & 434.97 \\ 0 & 286.27 & 399.74 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

We also retrieved the fisheye distortion coefficients k_l and k_r for the left and right cameras respectively.

$$k_l = \begin{pmatrix} -0.0127 & 0.0508 & -0.0491 & 0.01 \end{pmatrix} \tag{5}$$

$$k_r = \begin{pmatrix} -0.0153 & 0.0589 & -0.0595 & 0.0143 \end{pmatrix} \tag{6}$$

Next, we used OpenCV [3] to perform joint undistortion and rectification as illustrated in Fig. 3 with the Kannala-Brandt fisheye distortion model. Our new projection matrices for the left and right cameras are P_l and P_r respectively.

$$P_l = \begin{pmatrix} 286.29 & 0 & 436.76 & 0 \\ 0 & 286.29 & 336.08 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{7}$$

$$P_r = \begin{pmatrix} 286.29 & 0 & 436.76 & -18.17 \\ 0 & 286.29 & 336.08 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{8}$$

Figure 4. Sample rendered RGB stereo pair and corresponding depth frame from a hotel scene in Replica

5.2. Dataset

To ensure robust generalization, we generate our own synthetic dataset of RGB stereo pairs and depth images from existing high-resolution 3D ground truths. In particular, we choose to use the Replica dataset released by Facebook Reality Labs [20] for its 18 highly photo-realistic indoor scene reconstructions.

We used the intrinsics and extrinsics extracted from the T261 camera to render views from Replica. We first chose 4 scenes from each of the Replica categories: `hotel_0`, `apartment_1`, `office_2`, and `frl_apartment_3`. For each one, we rendered 100 RGB stereo pairs and ground depth along a circular trajectory in the middle of the scene at a height of approximately human eye-level, as illustrated in Fig. 4. We created a training set of 320 stereo and depth pairs, and left out the remaining 80 pairs for validation. We cropped our original images from 800 x 848 pixels to a square, and then resized to 432 x 432 pixels via bilinear interpolation. We obtained disparity from the rendered depth for training by the following relation, then scaling by a maximum depth of 40,000 and a minimum depth of 4,000.

$$Z = \frac{Bf}{d} \tag{9}$$

Z represents the depth in meters, B represents the baseline in pixels, f is the focal length (after undistortion) in pixels, and d is the disparity ranging from 0 to 64.

5.3. Experiments

We implemented our approach in JAX [2] and Flax [10]. We trained our model over our synthetic training set using Adam optimization with a learning rate of 0.0003 and a batch size of 8. We report the end-point error (EPE) and 1-pixel error for the validation set; EPE is the mean disparity error and pixels and the 1-pixel error is the percentage of pixels where the disparity error is greater than 1-pixel.

5.4. Baseline

As a baseline comparison, we used OpenCV's StereoBM and StereoBinarySGBM block matching methods [3]. StereoBM computes the simple Sum of Squared Differences (SSD) to find disparity by matching blocks between

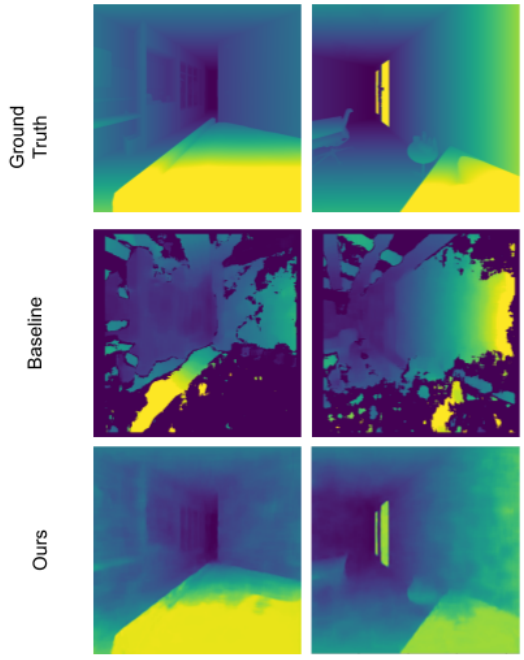

Figure 5. Qualitative comparison between our learned method and the baseline.

the left and right stereo images. On the other hand, StereoBinarySGBM uses a modified H. Hirschmuller algorithm [11] and performs semiglobal matching (SGM).

For the StereoBM algorithm, we used the default block size of 21 and a maximum disparity of 16. For the StereoSGBM algorithm, we use a minimum disparity of 0, maximum disparity of 16, block size of 16, uniqueness ratio of 10, speckle window size of 100, maximum disparity variation of 32, P1 (first parameter of disparity smoothness) of 600, P2 (second parameter of disparity smoothness) of 2400, and a maximum allowed difference (in integer pixel units) in the left-right disparity check of 1. These parameters are recommended by Intel RealSense for doing stereo with the T265 camera (which is very similar to the T261, our target hardware).

5.5. Discussion

The training and validation convergence of our model's loss is depicted in Fig. 6, and sample comparisons with the ground truth are available in Fig. 7.

We report a quantitative comparison error comparison between our method and the block matching baselines on the training set and the validation set in 5.5. In addition, Fig. 5 compares predicted disparities on several example pairs from our validation set.

We note that our method is able to broadly capture the main objects and shapes present in the scene, but struggles with capturing the finer outlines and details of smaller ob-

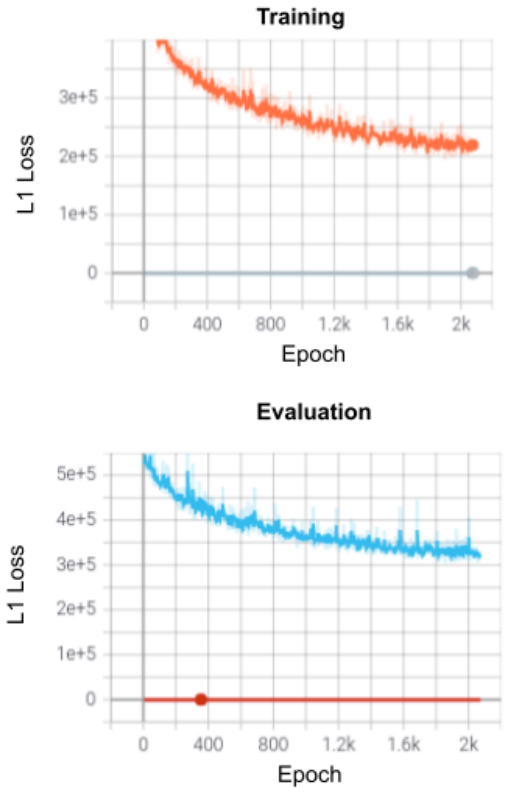

Figure 6. Our learned model convergence over the training and evaluation sets.

Method	EPE	1 px
Ours (Train)	1.07	0.38
Ours (Evaluation)	1.68	0.48
StereoBM (Evaluation)	11.61	0.92
StereoSGBM (Evaluation)	15.35	0.92

Table 1. Comparison of end-point error (EPE) and 1-pixel error

jects. For instance, the outline of the drawer space on the left and the finer features of the table on the right are not accurately captured by our method. In addition, for low-texture regions such as the wall or the bedsheets, there is undesired variation in the disparity prediction, shown by the patches seen in our model's outputted visualization. However, we note that this is better than the block-matching baseline which is incapable of matching certain regions or outputting a confident prediction for such regions at all.

6. Conclusions

In this project, we implemented a learned stereo matching network and compared our results to a classical block matching baseline. We conclude that by replacing the traditional components of the stereo matching pipeline with

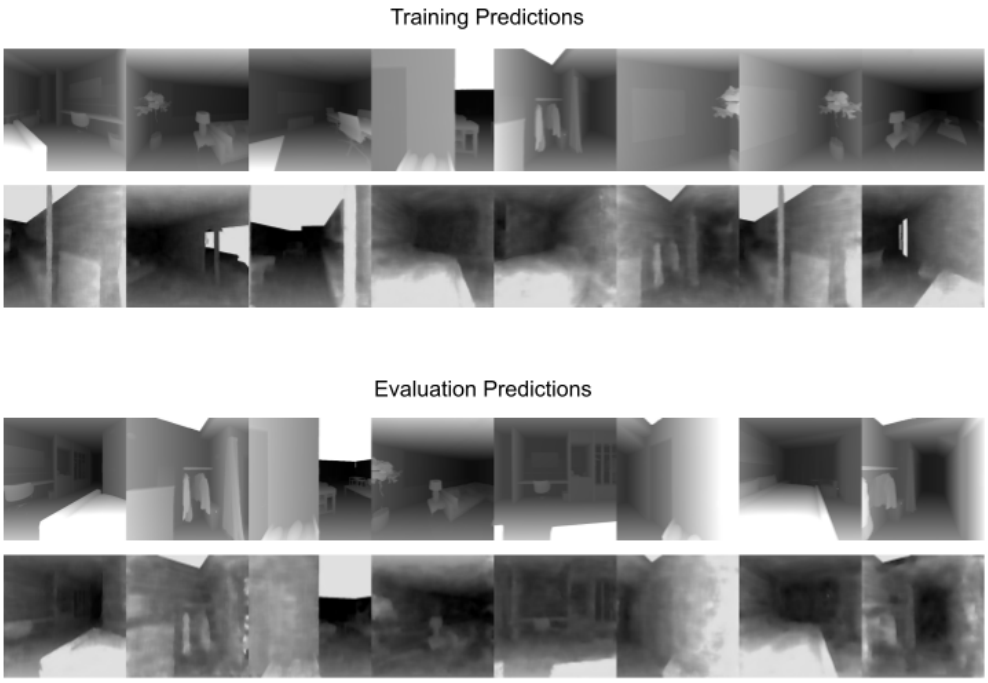

Figure 7. Comparison between a predicted disparity and ground truth on a training and validation batch.

differentiable counterparts in the form of deep neural networks, we are able to achieve a more robust depth estimation model as evidenced by our results on our synthetic dataset.

6.1. Limitations and Future Work

Our model has a fairly limited architecture compared to current state-of-the-art work. For instance, current state-of-the-art methods construct feature pyramids [15] in the feature extraction step to take advantage of cross-scale interactions. In addition, since Flax did not have a deformable convolution layer available, we to write our own implementation in Python which ran approximately an order-of-magnitude slower than a typical CUDA implementation. It would also worthwhile to analyze intermediate outputs of our model. For example, our learned deformable layers could indicate most useful neighbors in the aggregation process, providing interpretability into the behavior of our model. Lastly, we would need to run our learned model on the actual Intel T261 hardware to benchmark real-world performance, and use a dramatically expanded training set to increase generalization of performance.

7. Acknowledgements

The authors would like to thank the course staff for instruction during another difficult virtual quarter. We would also like to acknowledge our mentor Brent Yi for his guidance and feedback throughout the process.

References

[1] M. S. Banks, D. M. Hoffman, J. Kim, and G. Wetzstein. 3d displays. 2016. 1

[2] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 4

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 4

[4] R. Chabra, J. Straub, C. Sweeney, R. Newcombe, and H. Fuchs. Stereodrnet: Dilated residual stereo net, 2019. 3

[5] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018. 2

[6] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2

[7] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation, 2018. 2

[8] R. Garg, N. Wadhwa, S. Ansari, and J. T. Barron. Learning single camera depth estimation using dual-pixels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7628–7637, 2019. 2

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 2

[10] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee. Flax: A neural network library and ecosystem for JAX, 2020. 4

[11] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008. 5

[12] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, 2013. 2

[13] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. *CoRR*, abs/1703.04309, 2017. 2, 3

[14] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression, 2017. 2

[15] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 2, 6

[16] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016. 2

[17] G. Papandreou, I. Kokkinos, and P. Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 390–399, 2015. 3

[18] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE, 2020. 1

[19] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, 2001. 2

[20] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 4

[21] H. Xu and J. Zhang. Aanet: Adaptive aggregation network for efficient stereo matching. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1956–1965. IEEE, 2020. 2

[22] L. Zou and Y. Li. A method of stereo vision matching based on opencv. *2010 International Conference on Audio, Language and Image Processing*, pages 185–190, 2010. 2