Survey of Disparity Map Algorithms Intended for Real Time Stereoscopic Depth Estimation

Erick Blankenberg
Stanford University
Stanford, CA
eblanken@stanford.edu

Scott Blankenberg
Stanford University
Stanford, CA
sblanken@stanford.edu

Abstract

Autonomous vehicles rely on depth estimates derived from several sensor modalities for navigation and object detection. Stereoscopic depth estimation infers real-world distance to objects under a given target pixel from the displacement of the corresponding pixel in the other image of the stereo pair and known baseline distance between cameras. This approach provides dense depth estimates with inexpensive equipment that can be used to augment cheaper sensors to reduce overall system cost while maintaining good performance.

However, estimating the correspondence of pixels between images in stereo pairs to establish relative displacement is computationally expensive. While traditional real-time disparity algorithms typically sacrifice accuracy for speed, new deep-learning models promise to retain accuracy while achieving real-time performance on dedicated hardware.

This article evaluates two such algorithms, DeepPruner from Uber Research and the Pyramid Stereo Matching Network by Chang et al, which have ranked well in the KITTI Stereo competition. The traditional Stereo Block Matching Algorithm packaged in OpenCV is used as a baseline for comparison. It was found that all three algorithms performed well on simple stereo pairs from the 2001 dataset while DeepPruner performed the best on complex scenes followed by PSMNet.

1. Introduction and Motivation

Autonomous vehicles require highly detailed and frequently updated point cloud representations of their environment to safely navigate in a dynamic environment at high speed. This data is typically acquired through the use of redundant LIDAR, stereoscopic camera, and RADAR subsystems whose results are integrated via sensor fusion.

The survey performed by Yurtsever et al. [1] provides an overview of the characteristics of each of these sensors. Notably LIDAR provides accurate range estimates but sensors capable of producing dense point clouds are extremely expensive. LIDAR was initially favored over stereoscopic cameras as the deployment of stereoscopic cameras was hindered by the lack of availability of cheap high resolution cameras capable of operating at low light levels and by the computational cost of computing depth estimates. These issues have been addressed in recent years enabling stereo systems to achieve adequate performance for autonomous driving tasks at low cost [2]. As imaging technology and computational power have improved, interest in the development of efficient and fast methods for depth estimation suitable for autonomous driving tasks has increased.

Stereoscopic depth estimation infers real-world distance to objects by finding the relative displacement of corresponding pixels representing the same object in both stereo pairs. Once this disparity is established, the real world distance to the object can be inferred via triangulation. However, efficiently and accurately identifying corresponding pairs of pixels is a long-standing area of research in computer vision.

Traditional approaches estimate depth by identifying robust features in both images and computing disparity estimates which aligns features in both images guided by algorithm-specific heuristics. Algorithms differ in what features are considered, how the search for corresponding pixels is conducted, and how global context or neighboring pixels inform disparity selection. The best traditional approaches suffer when trying to estimate disparities for occluded pixels, plain areas, thin objects, and repeated textures. Real-time performance is achieved with a significant sacrifice in accuracy [3].

However, recent advancements in deep-learning based approaches promise to deliver excellent performance at speeds suitable for real-time applications. Here we evaluate two such algorithms, DeepPruner and PSMNet.

The DeepPruner algorithm introduced by Duggal et al. in 2019 achieved first place or near first place in accuracy in

several challenges while evaluating stereo pairs eight times faster than the then contemporary state-of-the-art approach at only 62ms per frame when running on a GPU [4]. The PSMNet algorithm developed by Chang et al. achieved good results in the KITTI 2015 leaderboard. We also included the standard StereoBM algorithm written by Kurt Konolige packaged in OpenCV. An overview of each of these algorithms will be provided in the **Methods** section below.

2. Related Work

In many papers proposing new algorithms authors compare the performance of the new approach to that of prior state-of-the-art algorithms. Other papers are exclusively dedicated to collecting and comparing many contemporary algorithms as far back as 1994 [5].

In recent years academic institutions have hosted sites such as the Middlebury Stereo Vision page and the KITTI Benchmark Suite [6], [7] which provide training data-sets along with public rankings of submitted algorithms evaluated on a hidden test set. These datasets have become standard benchmarks for disparity estimation algorithms.

2.1. Middlebury Dataset

The Middlebury Dataset has served as a popular benchmark for evaluation of disparity estimation algorithms since 2001. The site provides public stereo pairs depicting indoor scenes for use in research and retains a private collection which is used to rank algorithms submitted through the provided API. The publically available list of ranked algorithm contains 145 entries evaluated on the most recent dataset and includes metrics such as run-time and average absolute error on each of the images [6].

2.2. KITTI

The KITTI Stereo Vision Benchmark was introduced in 2012 and provides datasets and rankings for submitted algorithms in several tasks relevant to autonomous driving including stereo depth estimation, odometry, and object tracking. The data is generated by the Karlsruhes Institute of Technologie's Annieway autonomous driving platform on public roads which, as the site notes, is significantly more challenging than the laboratory setting of the Middlebury datasets [7].

3. Methods

An overview of the data used for this evaluation and the algorithms employed is provided below.

3.1. Data

Stereo pairs from the 2001 and 2014 Middlebury Data Sets were collected for evaluation [6].

The 2001 dataset consists of synthetic images composed of intersecting textured planes. Below the left image and ground truth disparity for "Poster" is shown.

Figure 1: barn2 data

The 2014 dataset consists of real-world images composed of arrangements of household objects. Below the left image and ground truth disparity for "Jade plant" is shown.

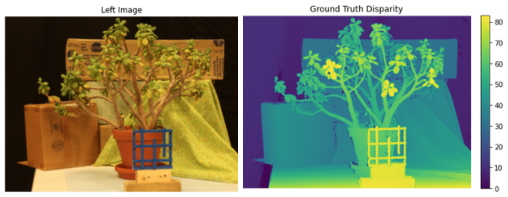

Figure 2: jade plant data

Prior to assessment camera calibration matrices were applied to images when provided and the images were scaled to fit within a bounding box 430 pixels tall and 380 pixels wide. Scaling was employed so that all images would have features of comparable size and so that our computers would not run out of memory while running DeepPruner and PSMNet. The shape and values of the corresponding ground truth disparity maps were scaled by the same factor.

3.2. DeepPruner

The DeepPruner network is a fully differentiable neural network designed to drastically decrease the time it takes to compute depth maps for stereo images. DeepPruner is based on two main ideas. First, instead of directly computing the entire cost volume for each left pixel and then finding the pixel in the corresponding epipolar line which has the lowest cost, it is better to try to prune out many candidate disparity values as soon as possible. Second, neighboring pixels in real world images tend to share similar disparities. To take advantage of this observation, DeepPruner attempts to propagate disparity information amongst neighboring pixels.

DeepPruner first uses a feature extraction network to transform both the right and left images into useful repre-

sentations. The feature network used is a 2D convolutional neural network which has spatial pyramid pooling. The final feature maps are one quarter the size of the original images.

These features are then feed into the differentiable Path Match algorithm and a sparse representation of the cost volume is produced. The original Patch Match algorithm was designed to create dense correspondences among images and took advantage of the fact that neighboring pixels have similar disparities. The creators of DeepPruner build upon Patch Match by unrolling it as a recurrent neural network. In this way the module becomes fully differentiable. The first layer to this recurrent neural network is a particle sampling layer which randomly creates k disparity values for each pixel. Then the propagation layer convolves the particles coming from close by pixels. Finally, the evaluation layer takes each pixel and correlates it with candidate disparity features. For each pixel, the top k disparity candidates are carried into the next iteration or time step of the recurrent network.

The sparse cost volume disparities created by the Patch Match network are then feed into a confidence prediction network along with the left and right images. For each pixel, the network learns a range over which it is confident the given pixel's disparity is in. This confidence prediction network follows an encoder-decoder architecture. This confidence range allows DeepPruner to prune out unlikely disparity values and hence avoid calculating a full set of matching costs.

The final module is a 3D convolutional neural network which takes in the left image, the right image, and the sparse cost volume as inputs. The output is a list of disparities over the confidence range of each pixel. Softmax is then applied to predict the actual disparity from the list of disparities for each pixel [3].

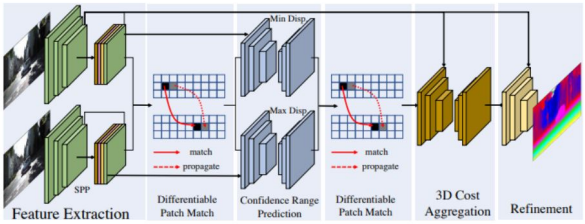

Figure 3: Deep Pruner Architecture [3]

3.3. Pyramid Stereo Matching Network

The creators of the Pyramid Stereo Matching Network(PSMNet) attempt to efficiently utilize global context information by using a learning methodology which does not involve any post-processing.To begin, the left and right images are fed into two separate convolutional neural networks which share weights. These networks learn pixel-level feature representations of each image which are then fed into the spatial pyramid pooling module(SPP). The SPP learns how to generalize the pixel-level input features into regional-level features which incorporate different scales of receptive fields. The regional level features for each image are then fed into two separate convolutional neural networks which share weights. The output of each of these networks are then concatenated into a cost volume. The cost volume is then fed into a 3D convolutional neural network with a stacked hourglass architecture. This 3D network then regularizes the cost volume in order to further incorporate even more global context into our cost volume. Finally the cost volume is upsampled and then passed through a disparity regression function which outputs the final disparity map [8]

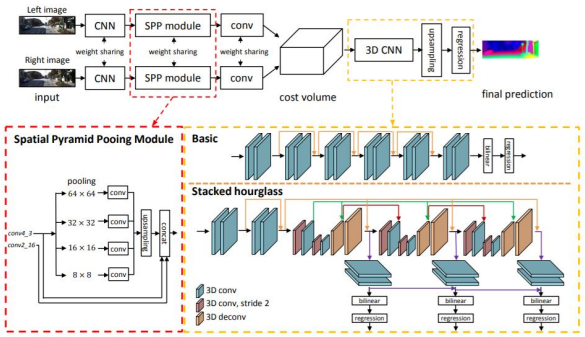

Figure 4: PSMNet Architecture [8]

3.4. Stereo Block Matching Algorithm

For our baseline, we use block matching, which is a relatively simple and fast window based local matching method. The algorithm first applies a Sobel prefilter to the input images to reduce noise [9]. Then for each pixel in the left input image, the algorithm scans its corresponding epipolar line in the right image to look for the best match. More specifically it considers a window around the pixel of interest in the left image and compares this window of pixels to the windows surrounding pixels along the epipolar line in the right image [10]. See figure 5 for an illustration. There are several cost functions one may use to compare the windows such as sum of square differences, maximum of absolute differences, and the sum of absolute differences. For the implementation in this paper we choose the sum of absolute differences metric(SAD), since it has been shown to give a good trade off between accuracy and speed [9].

For a given disparity d, the SAD match between a pixel at (x_0, y_0) in the left image and a pixel (x_0+d, y_0) in the left image is given by

$$Cost(d) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} |IL_{x_0+i,y_0+j} - IR_{x_0+d+i,y_0+j}| \tag{1}$$

where our window is 2N+1 by 2N+1 and $IL_{x,y}$ is the intensity at pixel (x,y) in the left image and $IR_{x,y}$ is the intensity at pixel (x,y) in the right image.

Two important parameters to the block matching algorithm is the minimum disparity, which controls where the scan line search begins in the right image and the number of disparities, which controls over how many pixels the search is conducted over. Another important parameter to set is the size of the windows being compared. Having a small window size produces sharper edges but leads to more noise, while larger windows lead to smoother outputs. Once a epipolar line is scanned, block matching uses the Winner Take All principle to choose the disparity. Thus the pixel in the right image which minimizes the SAD between its window and the window of the given pixel in the left image is chosen as the disparity. Note that this approach to choosing pixels is simple yet very fast. [10].

Since it does not use global context like the deep methods do, block matching often fails at discontinuities and in plain regions. Moreover, block matching is sensitive to noise and lighting conditions in the images [10]. Another weakness of block matching is that the aforementioned parameters often have to be hand-tweaked based on image size and composition. In our use of the algorithm we noticed that the optimal set of parameter values often depended upon which dataset the image was pulled from. Nonetheless, block matching is a standard high speed stereo matching algorithm which provides a good benchmark to test our two deep methods against.

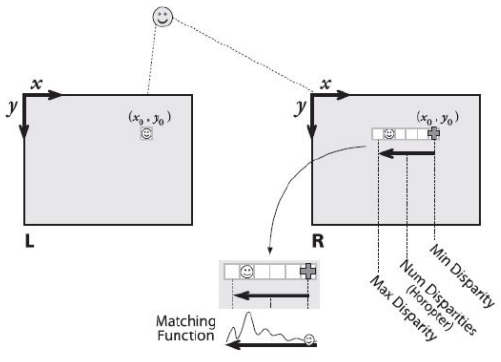

Figure 5: Block Matching [9]

4. Results and Analysis

Results of the evaluation of the algorithm on the Barn2 image from the 2001 dataset and the Jade Plant image from 2014 are presented below for qualitative analysis. The performance of each algorithm for all 8 images in the 2001 dataset and 25 images from the 2014 dataset is also presented.

4.1. Sample Evaluation and Qualitative Analysis

For illustration the performance of all three algorithms on the previously mentioned Barn2 and Jade Plant stereo pairs in terms of absolute disparity error and proportion of pixels whose error is less than the given threshold is shown below.

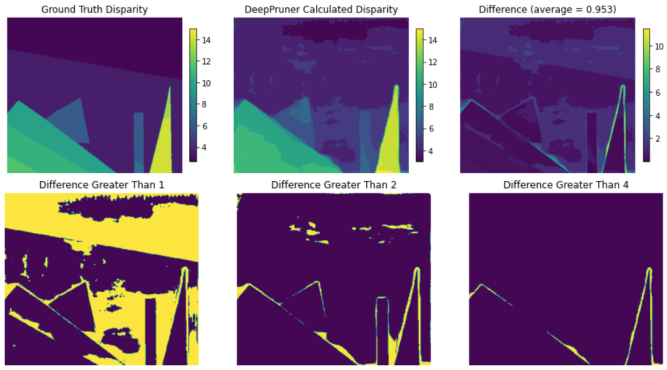

Figure 6: DeepPruner disparity error and thresholds on Barn2

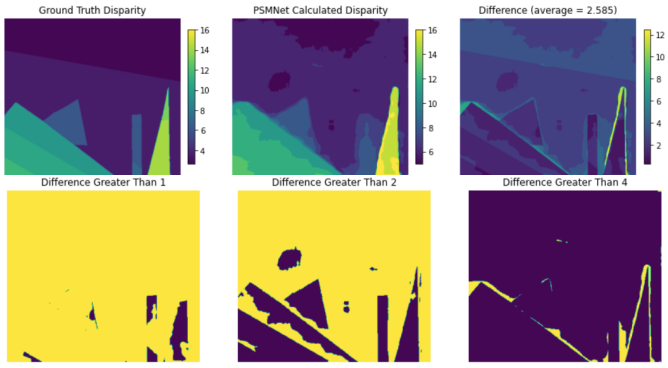

Figure 7: PSMNet disparity error and thresholds on Barn2

In the images for Barn2 all algorithms provide dense depth estimates and pixels whose disparity error is greater than 4 pixels are sparse. The disparity error is influenced by quantization of the ground truth disparity values as seen in the banding patterns on planes in the ground truth disparity image and in similar error bands most clearly visible in the Threshold Greater than 2 plot for DeepPruner. All algo-

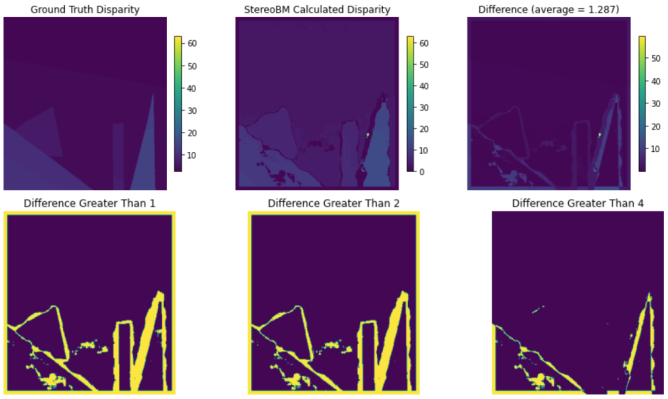

Figure 8: StereoBM disparity error and thresholds on Barn2

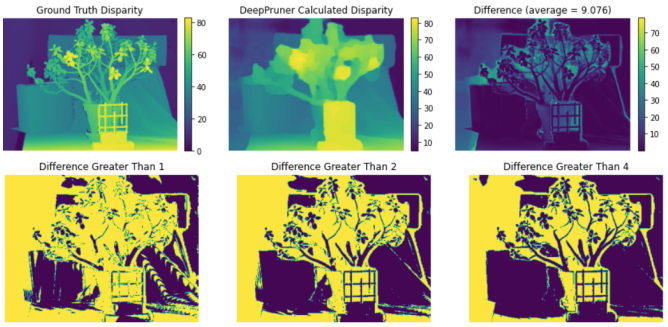

Figure 9: DeepPruner disparity error and thresholds on Jade Plant

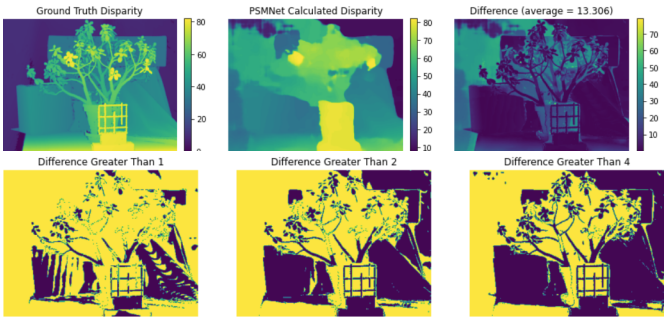

Figure 10: PSMNet disparity error and thresholds on Jade Plant

rithms appear to overpaint disparity estimates for closer objects which results in a halo like effect in the difference map. StereoBM performs adequately but is sensitive to noise, in this image it assigned a very high disparity value to a few anomalous pixels.

In the Jade Plant stereo pair, only the deep-learning models are able to provide adequate estimates of disparity. Whereas Barn2 was composed of large flat planes with smooth gradients, Jade Plant has several thin features which are challenging for all algorithms. StereoBM fails to pro-

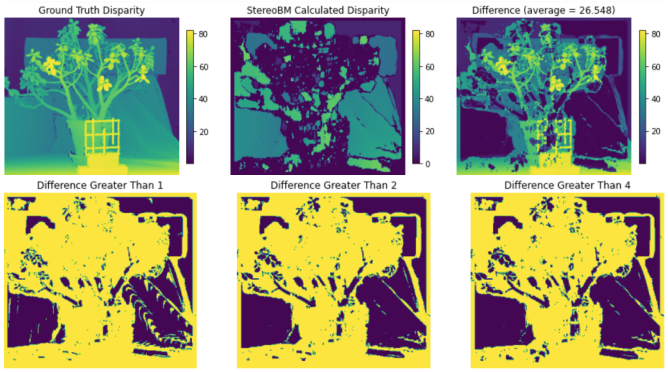

Figure 11: StereoBM disparity error and thresholds on Jade Plant

vide estimates in many areas where the plant is located but has adequate performance on the boxes and sheets in the background. This matches observations on other images from the 2014 dataset where StereoBM will fail to provide an estimate in regions with complex geometry and thin features but can handle flat planes acceptably. We find also that StereoBM will assign extremely large disparity values to some pixels in these regions which further contributes to error. Though the deep-learning models provide depth estimates in the region with the plant, the previously discussed overpainting causes significant loss of detail in this area. DeepPruner performs the best in this regard not only in the Jade Plant image but in all of the images from the 2014 dataset.

4.2. Dataset Performance and Quantitative Analysis

The performance of all three of these algorithms on all images from the 2001 and 2014 datasets in terms of average absolute disparity error and pixels whose absolute error is larger than the given threshold are shown in the following figures. We found that DeepPruner had an averag error 0.84 pixels of disparity on the 2001 dataset and 3.05 pixels on the 2014 dataset. PSMNet had an average error of 2.11 pixels on the 2001 dataset and 3.90 pixels on the 2014 dataset. StereoBM had an average error of 1.76 pixels on the 2001 dataset and 8.59 pixels on the 2014 dataset.

We can see that all algorithms perform well on the 2001 dataset with DeepPruner in the lead. However, for the more complex 2014 dataset, there is a much larger distribution of performance. DeepPruner performs the best consistently with PSMNet following closely behind. StereoBM has a broad distribution of results in many cases where more than 40% of pixels have a disparity error greater than 4. Both DeepPruner and PSMNet usually have less than 20% of pixels with an error greater than 4.

We have also included timing information for our runs

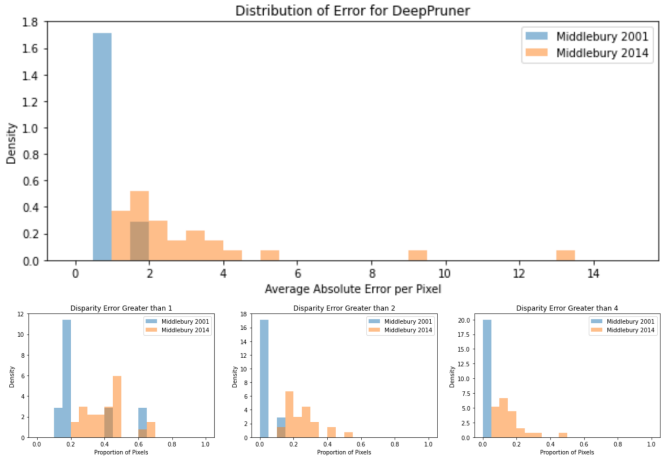

Figure 12: DeepNet disparity error and thresholds distribution by dataset

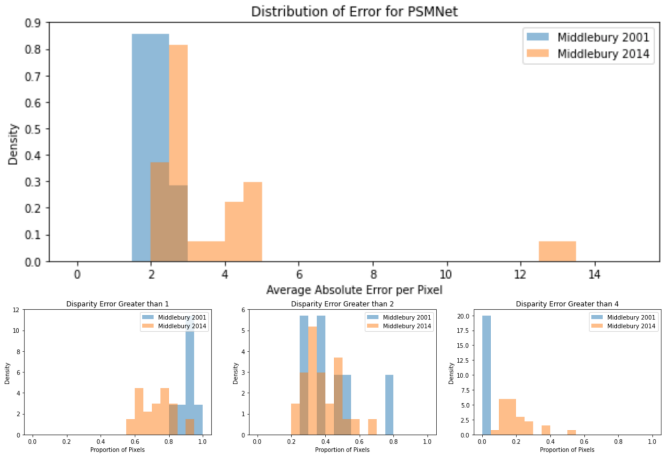

Figure 13: PSMNet disparity error and thresholds distribution by dataset

but note that as these were evaluated on a CPU rather than a GPU the deep-learning models are significantly slower than the times achieved in the original papers. We found that DeepPruner had an average runtime of 27.17 seconds, PSMNet 56.50 seconds, and StereoBM 0.0101 seconds. These tests were conducted on a 2.9 GHz Intel Core i5 in a 2015 Macbook Pro with no CUDA acceleration.

5. Conclusion

We evaluated DeepPruner, PSMNet, and StereoPSM, on 9 images from the 2001 and 2014 Middlebury Datasets. We found that all three algorithms performed well on the simple 2001 images but that only PSMNet and DeepPruner provided consistent results with the more complex 2014 compositions. DeepPruner had the strongest performance on both datasets.

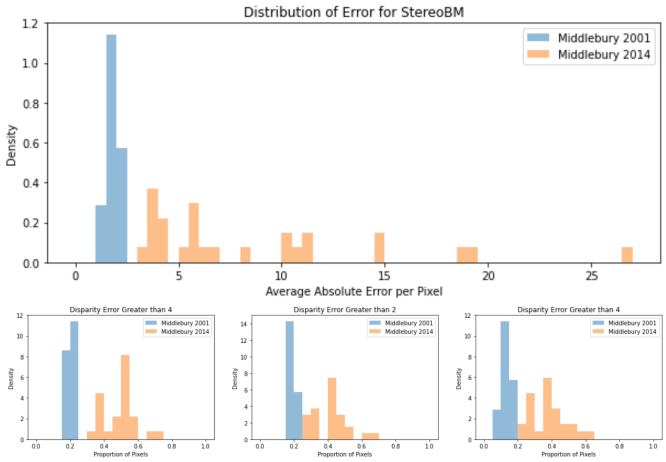

Figure 14: PSMNet disparity error and thresholds distribution by dataset

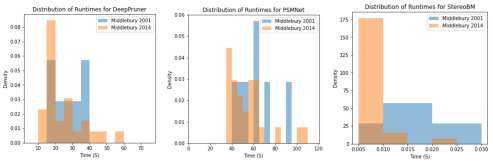

Figure 15: StereoBM disparity error and thresholds distribution by dataset

5.1. Limitations and Future Works

Given more time we would like to test each method on outdoor scenes from the KITTI database. Recall that the task of creating disparity maps for these outdoor images is generally harder than that of creating disparity maps for synthetic images from Middlebury. Moreover, for many autonomous applications, outdoor scenes are probably more relevant. These outdoor sets also include occlusion maps which would allow for evaluation of performance in regions that appear in only one image of the stereo pair and must be inferred.

We would also like to implement a series of methods not based on deep learning intended for real-time application from the KITTI and Middlebury records page. Our goal would be to find a potential method which would have a very low memory footprint or is highly parallelizable with preformance substantially better than that of the Stereo Block Matching algorithm. One traditional method we had started to implement, but never got a chance to finish, was a modular fuzzy logic disparity algorithm suitable for hardware implementation [11].

6. Acknowledgements

We would like to thank Chang et al. for providing code and pre-trained models for StereoPSM [12], Duggal et al.

for providing code and pre-trained models for DeepPruner [4], and Kurt Konolige for his implementation of the Stereo Block Matching algorithm provided in OpenCV. We would also like to thank Julien Martel for serving as our advisor in this project and Professor Gordon Wetzstein for providing this opportunity through EE367 this quarter.

References

[1] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[2] https://www.ambarella.com/blog/a-closer-look-at-lidar-and-stereovision/.

[3] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4384–4393, 2019.

[4] https://github.com/uber-research/deeppruner.

[5] R Lane. N thacker," stereo vision research: An algorithm survey, 1994.

[6] https://vision.middlebury.edu/stereo/data/.

[7] http://www.cvlibs.net/datasets/kitti/.

[8] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[9] Bryan Hale Bodkin. Real-time mobile stereo vision. *Master's Thesis, University of Tennessee,*, 2012.

[10] TIANYU Gu. Real time obstacle depth perception using stereo vision. *A thesis presented to the graduate school of the University of Florida for Master of Science in University of Florida*, 2014.

[11] M Perez-Patricio, Abiel Aguilar-González, Jorge-Luis Camas-Anzueto, and Miguel Arias-Estrada. A fuzzy logic approach for stereo matching suited for real-time processing. *International Journal of Computer Applications*, 113(2), 2015.

[12] https://github.com/jiarenchang/psmnet.