Lab Vision Systems Project Report

Elif Cansu YILDIZ and Salih MARANGOZ

University of Bonn, 53113 Bonn, Germany

Abstract. In this lab report, we handle the stereo depth estimation task using different models proposed in a diversity of research papers. GC-Net [1] and PSMNet [2] are a few examples of model solutions that achieve good results. We solve this task by implementing these models' architecture including feature extraction, cost volume, 3-D convolutions, and regression parts. With all experiments, we surpassed the baseline mean loss and mean 3-pixel error on the KITTI 2015 test dataset with and without pre-training on Scene Flow. To attain the highest result, we tried different ideas and did hyperparameter tuning in a variety of experiments.

1 Introduction

Depth estimation plays a key role in scene understanding [3], augmented reality [4], self-driving cars [5], robotics, etc. Although using LiDAR for estimating depth is a popular way, the high costs and measuring sensitivity to weather conditions bring many disadvantages during outdoor experiments. In this regard, depth estimation with stereo camera, also called stereo matching, is an alternative approach used in state-of-the-art algorithms. The main idea of stereo matching is finding a corresponding pixel on the right camera frame for each pixel on the left camera where two cameras have horizontal displacement. Therefore, the depth of the point can be estimated by knowing the distances between the pixels of stereo cameras and camera parameters.

In this project, the main objective is to compute the disparity, which is an inversely proportional calculation of scene depth, of each pixel using stereo pair of camera images. A simple approach to disparity estimation consists of a few steps [6]. The first one is, extracting the features from both cameras to obtain more valuable information rather than raw color values, thus improving color matching. The second step is forming the cost volume to estimate matching of the left and right feature maps across each disparity level. This allows to learn to incorporate context which can operate over feature maps. The last step is matching features and estimating the disparity with the computed cost volume using 3-D convolutional and regressor modules, thus acquiring a disparity prediction for each input pixel. The number of steps can be extended further depending on the task.

The rest of the report is organized as follows; Section 2 describes the datasets. Section 3 explains the baseline architecture and the models (GC-Net, PSMNet) in detail. Section 4 focuses on experimental details while section 5 presents the

results of each experiment and discusses the influences of each improvement ideas. In section 6, the final comments are concluded. Section 7 reflects the contribution of each author.

2 Dataset

We trained and evaluated the models using two stereo datasets:

1. Scene Flow [7]: A large synthetic dataset including Driving, FlyingThings3D, and Monkaa subset with 540×960 size (height $\times$ width). 1600 pairs including 15mm focal length, scene backward and forwards from Driving subset, 26760 scenes consisting of all test and train samples from FlyingThings3D, and 8664 pairs of images from Monkaa have been used, which makes a total of 37024 data. All subsets provide dense disparity maps as ground truth. Since some pixels have negative disparities, we excluded them in the loss computation for all experiments. Therefore, disparity values start from 0 and go beyond 500 pixels.

2. KITTI 2015 [8, 9]: A real-world dataset with a street view from a driving car. It consists of 200 training stereo RGB image pairs and sparse ground truth disparities obtained using Velodyne LiDAR. Other 200 test images are not shared with the public but used for ranking of different models in the leaderboard. Image sizes are mostly H=375 and W=1242 and have similar resolutions in some images. We clipped up to 10 pixels from the top and up to 15 from the sides when the input size is not appropriate for the model. We used 150 images for training and validation, 50 images for evaluation. We further divided the 150 images into training set for fine-tuning (80%) and validation set (20%). Disparity values range between $[0-256]$ pixels and only 20% of disparity values are valid because of the sparsity of disparity maps. Additionally, random jittering is applied to the training set as an augmentation method.

3 Network Architecture

The model is expected to learn an end-to-end mapping from stereo images to disparity maps using deep learning. Basically, the model comprises three submodules namely feature extraction, joined processing, and disparity regression parts. The simple model architecture in this project is shown in Figure 1.

In the feature extraction module, weights are shared between left and right feature extractors, and the cascade of several residual blocks is used. The goal of this module is to learn a deep representation of raw pixel intensities. Therefore, it ensures more robustness to ambiguities and being able to include local context information. After acquiring feature maps of each pair of images from the feature extraction module, a cost volume is obtained by simply concatenating the left and right unary features across each disparity level, resulting in a 4D volume (feature size $\times$ disparity $\times$ height $\times$ width). The formula of the cost volume is presented in the Equation 1.

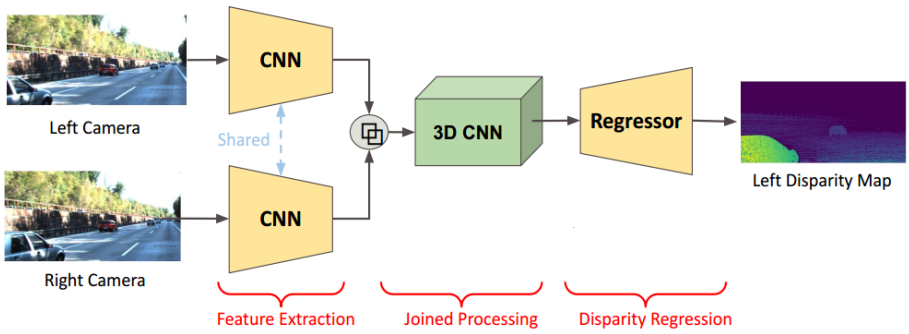

Fig. 1. Baseline Model Pipeline

$$E(D) = \sum_{x} C(x, d_x) \tag{1}$$

When using rectified stereo pairs, $C(x, d_x)$ measures the cost of matching pixel $x = (i, j)$ of the left image with the pixel $y = (i, j - d_x)$ of the right image. In this case, $d_x = D(x) \in [d_{min}, d_{max}]$ is the disparity at pixel x [10]. The joined processing part includes 3-D convolutional operations which process 3-D blocks on the cost volume and be able to learn feature representation from the height, width and disparity dimensions. This part is followed by upsampling back to size height × width × disparity. Finally, soft regression is performed to calculate the disparity map with size H × W. The formula of the predicted disparity ($\hat{d}$) regression [1, 2] is expressed in the Equation 2.

$$\hat{d} = \sum_{d=0}^{D_{max}} d \times \sigma(-c_d) \tag{2}$$

We experimented with two different architectures based on the baseline architecture (Figure 1) in this task:

3.1 GC-Net

The GC-Net [1] architecture is shown in Figure 2. The 2-D convolution part of the architecture contains eight residual blocks of combinations of 2-D convolutions and another convolutional layer afterward. Relu activation function and batch normalization are applied after each convolution in the residual blocks, but not after the last convolutional layer. Computation of the cost volume follows that and then Multi-Scale 3-D Convolutions including Encoder-Decoder idea in three dimensions is performed. $3 \times 3 \times 3$ convolutions in series are applied for each encoder level and 3-D transposed convolution for up-sampling is employed. We implemented the same architecture of the GC-Net except using the first 2 consecutive 3-D convolution layers instead of 3 in the 3-D convolution blocks.

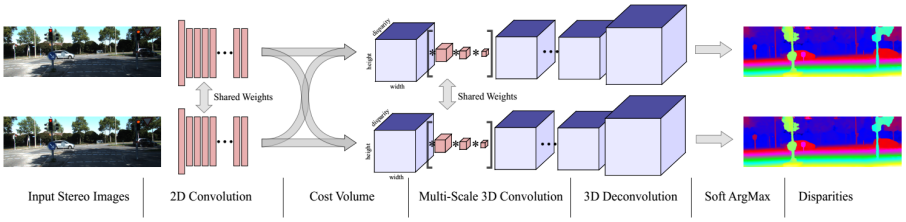

Fig. 2. GC-Net (Geometry and Context Network) learns an end-to-end mapping from a left/right image pair to disparity maps. Sequentially; both input images are processed separately with 2-D convolution feature extractors, computed features are used for forming the cost volume, multi-Scale 3-D convolutions are used for matching the cost volume features, intermediate outputs are upscaled with 3-D deconvolutions, and finally, soft-argmax is used for computing output disparities.

3.2 PSMNet

The PSMNet [2] is an end-to-end framework for stereo matching/depth estimation tasks using stereo pair of images to deal with the problem of lacking effective context information to find correspondence in occluded images. The architecture (in Figure 3), in addition to baseline architecture, comprises further ideas which are Spatial Pyramid Pooling (SPP) in the feature extraction module to combine features from different scales, and Stacked hourglass architecture in the 3-D convolutional part as an another option to the basic network. The Spatial Pyramid Pooling module contributes learning to incorporate hierarchical context information like the relationship between an object and its sub-regions. Moreover, involving different levels of features in the SPP module facilitates stereo matching. In the Stacked hourglass (encoder-decoder) architecture, more context information is learned. We implemented the same architecture of the PSMNet presented in the paper including SPP and stacked hourglass modules and used the open-source code [11] of the basic structure of PSMNet. We shall refer to the PSMNet model with SPP and stacked hourglass modules as Full-PSMNet, to the basic structure of the PSMNet as Basic-PSMNet throughout this paper.

4 Experimental Details

We mainly used PyTorch Deep Learning Framework [12] to implement train, validation, test and our models. Additionally, we used the Torchvision framework to define preprocessing steps and pre-trained feature extraction models. All models were end-to-end trained with Adam [13] optimizer ($\beta_1 = 0.9$; $\beta_2 = 0.999$). We performed color normalization on the entire dataset for data preprocessing [14]. The images for training and validation were randomly cropped to size H = 256 and W = 512 while image sizes of the test data of KITTI are cropped to the size H=352 and W=1216. Only the top of test images is cut for height whilst

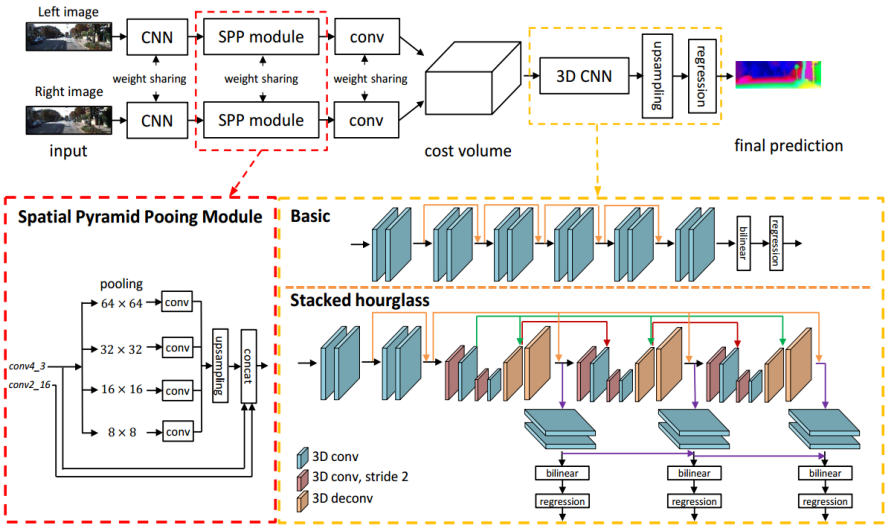

Fig. 3. PSMNet includes spatial pyramid pooling modules in the feature extraction and provides two options with basic and stacked hourglass 3-D convolutional building blocks. Different than the basic configuration, stacked hourglass configuration has three outputs for loss calculation used for training.

both sides of the width, from left and right, are cut. The batch size was set to 1 for the all the training, validation and test processes. The maximum disparity was set to 192 except one experiment. StepLR scheduler was used in some experiments. Concerning the parameters of the scheduler, we set gamma=0.75 and step size=1000 or 2500 depending on the experiment.

Regarding training the network, we pursued two different strategies on the GC-Net and PSMNet [1, 2]. The former is fine-tuning on KITTI using the pre-trained model on Scene Flow. The latter is training using only KITTI without pre-training.

For pre-training, we used the constant learning rate of 0.0001 for 10 epochs. Since the pre-training and fine-tuning datasets are of different sizes, we preferred to use the term iteration in the fine-tuning experiments instead of epoch to increase clarity. All fine-tunings were run for 20000 iterations. Conducted experiments for fine-tuning are listed below:

- Fine-tuning the Full-PSM model with different hyperparameters shown in the results section.
- Fine-tuning the Full-PSM model with learning rate of 0.0001 without any scheduler. The only difference of this experiment is strided cost volume part which is adding stride to the sliding operation while shifting and stacking the features obtained from the right and left images. This fine-tuning uses the pre-trained model of Full-PSM model with strided cost volume as described in subsection 4.4.

- Fine-tuning the Full-PSM model with Fully Convolutional ResNet50 [15] module as an extra feature extractor outputs concatenated to the fusion layer of the main feature extraction module, thus increasing the number of features from 320 to 2368.
- Fine-tuning the GC-Net model with different hyperparameters shown in the result section.
- Fine-tuning the Basic-PSM model with scheduler starting with the learning rate of 0.0001.

For training the models from scratch without using any pre-trained models, the Full-PSM model was trained for 20000 iterations with the learning rate of 0.0001. Two experiments, with and without the scheduler, were conducted. Also, we trained the Full-PSM model with the value of 96 for max disparity.

Experiments are being conducted on several machines with different configurations. One of them, Intel(R) Core(TM) i7-8700K CPU@3.70GHz with NVIDIA RTX3090 was able to process 3.3 samples per second. On this machine, Full-PSM pre-training steps took about 35 hours, and fine-tuning steps took about 4 hours with maximum disparity of 192.

In the following subsections, the evaluation metrics are presented in Section 4.1, followed by the analysis of selecting maximum disparity, handling of disparity values residing out of model's capability, and the explanation of strided cost volume idea, respectively.

4.1 Metrics

Considering the loss functions in training and evaluation, the smooth L1 loss is used as defined in Equation 3. We used $\beta = 1.0$ in all of our experiments.

$$SmoothL1(\hat{x}_i, x_i) = \begin{cases} 0.5 \cdot (x_n - \hat{x}_n)^2 \cdot \beta, & |x_n - \hat{x}_n| \leq \beta \\ |x_n - \hat{x}_n| - 0.5 \cdot \beta, & otherwise \end{cases} \tag{3}$$

For calculating the validation and test accuracy we used the 3-pixel error (3PE) metric as defined below. Equation 4 shows if predicted and true disparity values are correct (1) or not (0). This method applies strict 3-pixel error between 0 and 60 disparity values and increases the error tolerance starting from 60 to infinity. Calculating the loss for a single sample of dataset is shown in Equation 5 with N is the number of labeled pixels, x is the ground truth disparity and $\hat{x}$ is the predicted disparity. Non-valid disparity values are not shown in the equations and ignored in the computations.

$$CorrectDisp(\hat{x}_i, x_i) = \begin{cases} 1, & |x_n - \hat{x}_n| \leq 3 \\ 1, & |x_n - \hat{x}_n| \leq 0.05 \cdot x_i \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$3PE(X, \hat{X}) = 1 - \frac{1}{N} \sum_{i=1}^{N} CorrectDisp(\hat{x}_i, x_i) \tag{5}$$

Baseline metrics to achieve are 9% for the mean 3PE and 1.1 for mean SmoothL1 loss.

4.2 Choosing the Maximum Disparity

The maximum disparity affects the coverage of disparity pixels of training images and the prediction range of the model's disparity. Setting the maximum disparity too high leads to a larger cost volume, thus increasing training time and memory usage. To address this problem, as seen in Figure 4, we analyzed the maximum disparity parameter and coverage of the disparity values in the KITTI and Scene Flow.

Setting maximum disparity to 96 covered 99% of the disparity pixels in the KITTI. However, the coverage of Scene Flow with the value of 96 drops to 89%. Therefore, by choosing the value of 192 for the maximum disparity, we aimed to cover 97% of the disparity pixels of the Scene Flow. Additionally, we trained with 96 of maximum disparity on KITTI without pre-training on the Scene Flow in one of our experiments.

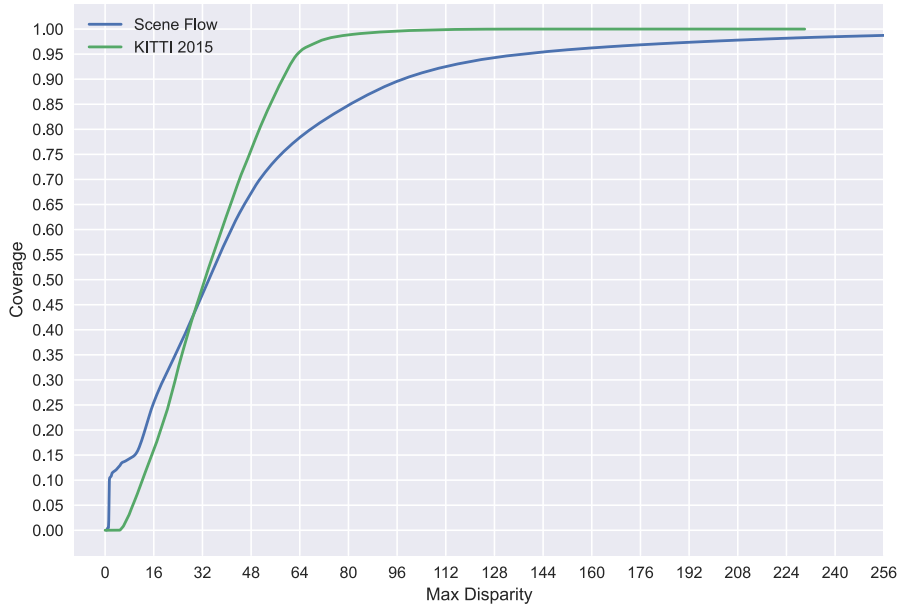

Fig. 4. Setting maximum disparity to 96 covers more than 99% of pixels in KITTI, but covers 89.5% of pixels in Scene Flow. So we preferred setting maximum disparity to 192 which covers 97.3% of pixels in Scene Flow.

4.3 Pixels with High Disparity Values

After setting the maximum disparity value to 192 in most of the experiments, we tried clipping and discarding the disparity values residing outside of this limit. With clipping, we enabled high disparity values to get involved in training. However, this method posed distortions in the outputs. Therefore, we discarded the disparity values which are outside of the limits in all experiments.

4.4 Strided Cost Volume

We noticed that cost volume calculation in PSMNet is done by shifting left and right feature pairs at most 1/4 of the maximum disparity value and that this method greatly improved the computational speed and used memory space. To move this method a little further, we tried to eliminate similar feature pairs in the cost volume calculation. To achieve this, we modified the default algorithm to add features in a strided way similar to [16]. When we set the stride value to 2, our model became 1.6x faster to train.

With the modification made, we think that the average accuracy decreased since the problem got more complicated for the modules after the cost volume. However this method can be handled in a different way, especially for self-driving cars. For example, by applying the stride operation at the sub-pixel level with resampling methods, more consistent measurements can be obtained for certain distances of the model.

5 Results

We conducted experiments with several settings to evaluate Full-PSMNet, GC-Net, Basic-PSMNet as listed in table 1. The best result is achieved with a 3.16% for 3PE error and 0.55 for mean loss by the Full-PSMNet including the learning rate scheduler starting with 0.0001. On the other hand, GC-Net could not surpass any of the results of PSMNet except the one that has the learning rate of 0.00001 as well as that lack pre-training. The worst result was acquired with the value of 8.24% for 3PE and 1.03 for mean loss by the Full-PSMNet comprising the setting of maximum disparity=96 and lacking pre-training. Even with the worst result we obtained, we surpassed the baseline values in all experiments.

When it comes to the results of the experiments of different models in table 1, all the fine-tuned Full-PSMNet results with different hyperparameter settings are in the range of 3.16% and 4.51% for the mean 3PE and of 0.55 and 0.71 for the mean SmoothL1 loss. Also, fine-tuned Basic-PSMNet has the outcome in those ranges. The Full-PSM with ResNet module also illustrated similar results in those ranges. Although GC-Net was not able to have better results than any PSMNet with pre-training for the mean SmoothL1 loss, all experiments of GC-Net outperformed the results of Full-PSMNet without pre-training for the mean 3PE. The best outcome of the GC-Net beat the result of Full-PSMNet with scheduler starting with learning rate 0.00001. We also tried strided cost volume

Model			Pre-Training	Hyperparameters			Evaluation Metrics	
Basic-PSM	Full-PSM	GC-Net		LR	Max Disp	Scheduler	3PE	Smooth L1
	✓		✓	10^{-4}	192		3.24%	0.5654
	✓		✓	10^{-5}	192		3.32%	0.5868
	✓		✓	10^{-3}	192	✓	3.77%	0.6287
	✓		✓	10^{-4}	192	✓	**3.16**%	**0.5473**
	$(✓)^{*}$		✓	10^{-4}	192	✓	3.41 %	0.6037
	✓		✓	10^{-5}	192	✓	4.51%	0.6978
	$(✓)^{**}$		✓	10^{-4}	192	✓	4.51 %	0.7081
	✓			10^{-4}	192		4.91%	0.7924
	✓			10^{-4}	192	✓	5.27%	0.8180
	✓			10^{-4}	96	✓	8.24%	1.0265
		✓	✓	10^{-3}	192		4.41%	0.7686
		✓	✓	10^{-4}	192		4.41%	0.7358
		✓	✓	10^{-4}	192	✓	4.62%	0.8087
✓			✓	10^{-4}	192	✓	3.48%	0.5889

Table 1. Test results of various experiments. Full-PSM with $(✓)^{*}$ indicates ResNet50 added feature extractor, Full-PSM with $(✓)^{**}$ indicates strided cost volume computation experiments.

in the Full-PSMNet. However, it did not provide a better outcome than the best result.

Considering the results of various hyperparameter settings, the learning rate of 0.00001 was always worse than 0.0001 regardless of including scheduler. Fine-tuning the GC-Net with lr=0.001 and lr=0.0001 without scheduler did not make any difference for 3PE (4.41% for both) and trying with scheduler was worse with 4.62% for 3PE. Using scheduler worked better only for the Full-PSMNet starting with 0.0001 which gives the best result in our experiments. For the other usages of scheduler underperformed than the same settings lacking scheduler.

After trying different configurations, we obtained the outputs of the best Full-PSM and GC-Net illustrated in Figures 5 and 6 and compared the outputs of these models in Figure 7. In general, we observed that GC-Net is able to perceive cables, pipes, street lambs as a whole. Although Full-PSMNet sometimes cannot succeed in perceiving them, we examined that it predicts disparity values including especially close objects better. Nevertheless, we think that failure of the detection of the objects like thin pipes might pose to accidents in self driving car applications. Additionally, we inspected that GC-Net and Full-PSMNet fine-tuned with learning rate scheduler predict higher disparity values for the sky in open fields, thus strongly suggesting preprocessing of noisy outputs before utilizing them in other algorithms. However, we did not face this problem with Full-PSMNet fine-tuned without learning rate scheduler as shown in the Figure 8.

We noticed that indecision exists in the windows of the cars in both datasets. While the data is being collected for KITTI 2015, the depth information did not include the interior of a car because the window of a car reflects in the LiDAR measurements. Sometimes these reflective surfaces do not appear on disparity

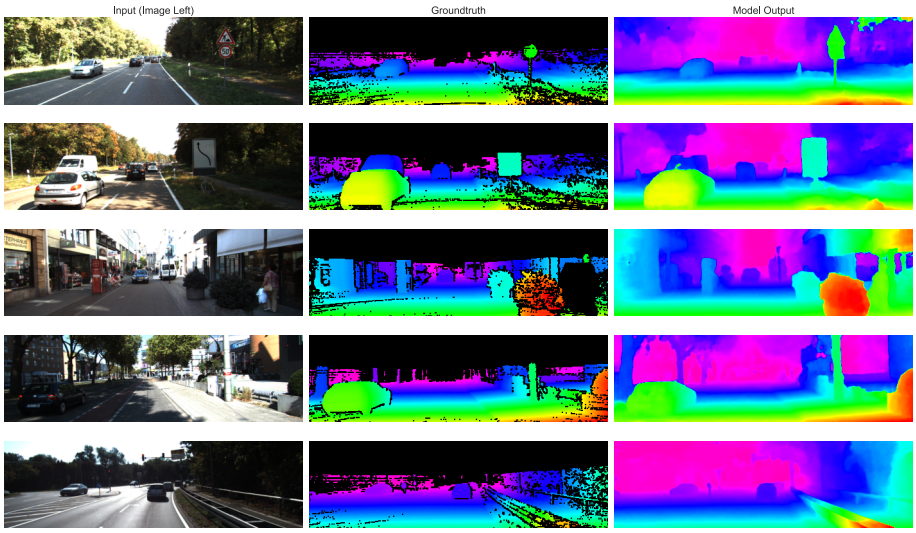

Fig. 5. PSMNet Outputs. Only left images are shown on the left, ground truth in the middle, and model outputs on the right. Different color codes encode different disparity values.

maps at all. This problem affects the GC-Net significantly since noticeable distortions occur in the shape of cars while the model tries to predict the interior of cars. We think that this tendency might have a harmful influence where reflection increases such as during rain, near to puddles, in front of glittery stores. To tackle this problem, we believe that pre-training the model with the VirtualKITTI [17, 18] including different weather conditions would be beneficial to obtain reliable results.

Apart from that, in KITTI, it can be seen that cropping LiDAR data to match with RGB Cameras resulted in sparse disparity maps. Solid state LiDARs are in development nowadays and have similar viewing angles to RGB cameras. Instead of using mechanical LiDARs, data recording with solid state LiDARs would help to create more dense and better disparity data.

6 Conclusion

In our study, PSMNet and GC-Net models achieved good results in stereo matching problems. However, obtaining disparity values in ill-posed regions still remains as a problem. We examined the impact of using the learning rate scheduler on PSMNet and observed that it can be detrimental. Also, we achieved lower success in our experiments than the PSMNet and GC-Net papers. When we examined this, we identified some differences (number of training epochs, hyperparameter settings, convolutional blocks) between the models presented in the paper and implementation.

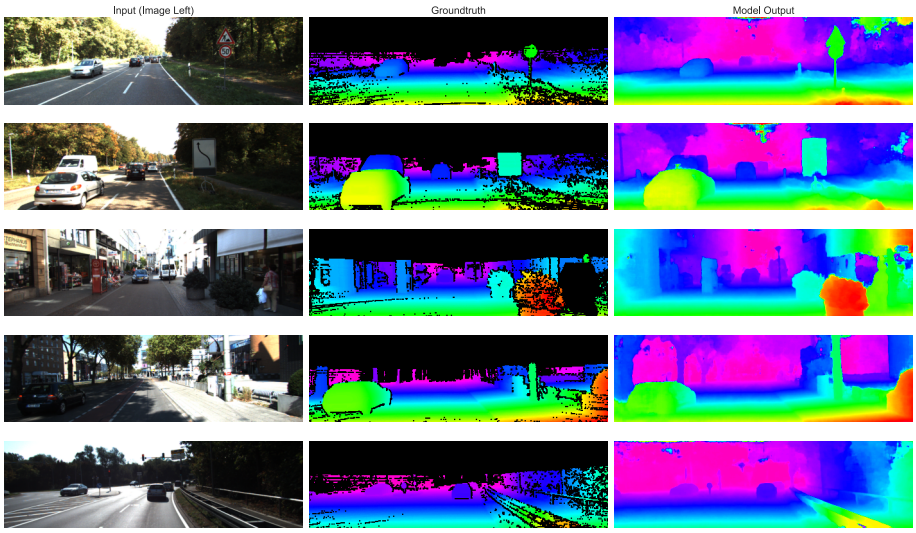

Fig. 6. GC-Net Outputs. Only left images are shown on the left, ground truth in the middle, and model outputs on the right. Different color codes encode different disparity values.

In this study, we also tried a strided version of the cost volume calculation. Since the modification made the model more complex, we had worse results, but we saw that the cost volume calculation can affect the speed and accuracy of the model significantly. For this reason, we think that left/right feature pairs can be matched to cost volume by learnable parameters, and this might further reduce computation resources.

7 Contributions

Regarding the source code; Salih contributed to implementing dataset loading & data preprocessing, implementing Full-PSMNet, pre-training of Full-PSMNet and Basic-PSMNet, implementing visualizations methods. Cansu contributed to implementing experiment mechanism, implementing training & evaluation, implementing GC-Net, pre-training of GC-Net, and fine-tuning all models. For Basic-PSMNet, source code [11] from GitHub is used. Considering all workload, the authors put the same amount of contribution to the project and report.

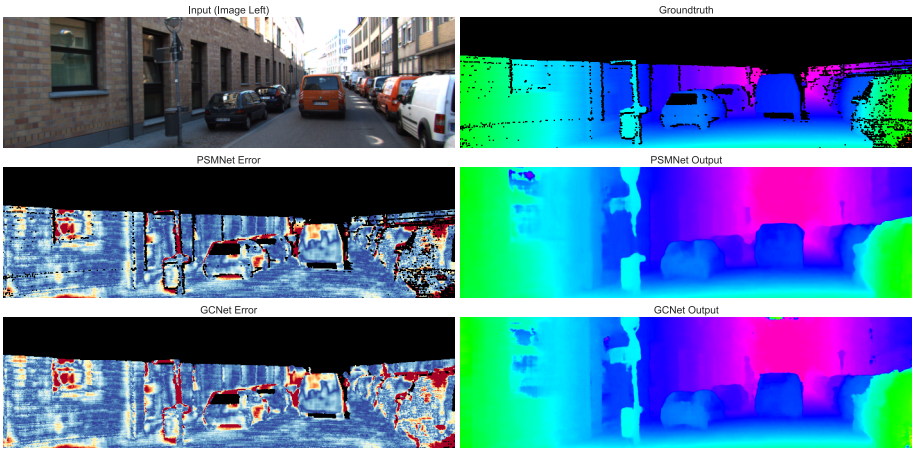

Fig. 7. GC-Net predicted disparity for the pipe better than PSMNet. On the error map, the red color defines 3 pixel error or more, blue color defines low error regions.

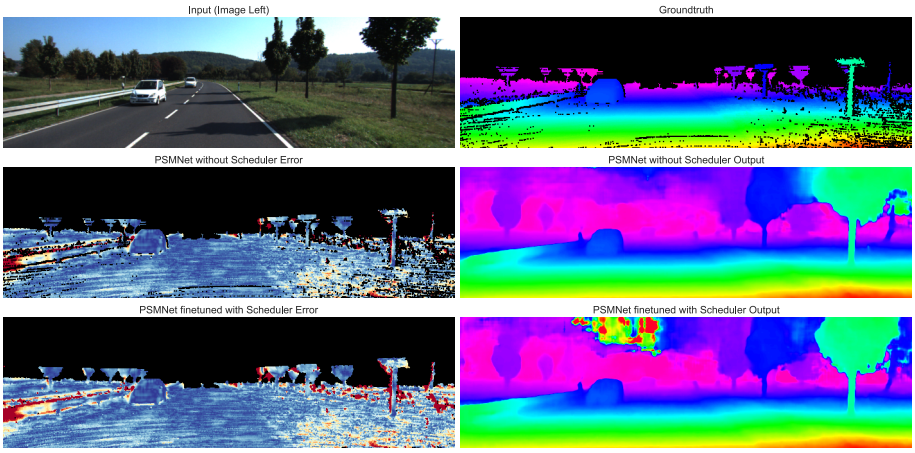

Fig. 8. Full-PSMNet fine-tuned with a learning rate scheduler detected a high disparity for the mountain, while Full-PSMNet fine-tuned without a learning rate scheduler was not affected by this problem. On the error map, the red color defines a 3 pixel error or more, blue color defines low error regions.

References

1. A. Kendall and et al., "End-to-end learning of geometry and context for deep stereo regression.," *arXiv preprint arXiv:1703.04309*, 2017.
2. J. R. Chang and Y. S. Chen, "Pyramid stereo matching network," *arXiv preprint arXiv:1803.08669*, 2018.
3. Z. Guo, Y. Huang, X. Hu, H. Wei, and B. Zhao, "A survey on deep learning based approaches for scene understanding in autonomous driving," *Electronics*, vol. 10, no. 4, 2021.
4. F. El Jamiy and R. Marsh, "Distance estimation in virtual reality and augmented reality: A survey," in *2019 IEEE International Conference on Electro Information Technology (EIT)*, pp. 063–068, IEEE, 2019.
5. C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz, *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.
6. D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision 47, 7–42*, 2002.
7. 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 *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.
8. M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.
9. M. Menze, C. Heipke, and A. Geiger, "Object scene flow," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.
10. H. Laga and et al., "A survey on deep learning techniques for stereo-based depth estimation," *arXiv preprint arXiv:2006.02535*, 2020.
11. J. R. Chang and Y. S. Chen, "Pyramid stereo matching network." https://github.com/JiaRenChang/PSMNet, 2021.
12. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.
13. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization.," *arXiv preprint arXiv:1412.6980*, 2014.
14. "Torch Vision Models." https://pytorch.org/vision/stable/models.html. Accessed: 2021-09-30.
15. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015.
16. C. Lu, H. Uchiyama, D. Thomas, A. Shimada, and R.-i. Taniguchi, "Sparse cost volume for efficient stereo matching," *Remote sensing*, vol. 10, no. 11, p. 1844, 2018.
17. A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.
18. Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," 2020.