Depth Reconstruction from Stereo Image Pairs

Xiuye Gu
xiuyegu@stanford.edu

Abstract

Depth information is a kind of important 3D information, and two images are the minimum requirement that we can obtain unambiguous depth information. So we work on the problem of depth reconstruction from stereo image pairs. We start from adopting sequence alignment algorithm to compute disparity, based on the similarity scores computed by mc-cnn [9]. According to our initial results, this pipeline does not work well compared with other mature pipelines, and the sequence alignment is prone to error. So we reimplement the state-of-the-art GC-Net [2], which regresses disparity in an end-to-end way. Experimental results reveal that, when training with groundtruth disparity masks, the model is prone to predict very blurry disparity maps, it works better when training without the masks. Our source codes reside in this repository: `https://github.com/laoreja/CS231A-project-stereo-matching`.

1. Introduction

The part in the course that attracts me most is the 3D geometry part. Most visual data we have are 2 dimensional, if we can reconstruct the 3D information, we will have better understanding of these data, since our world is 3 dimensional. Depth reconstruction is one of the 3D reconstruction tasks, and it has many applications in the real world, *e.g.*, autonomous robot navigation, mobility aid for visually impaired people, *etc*. Two images are the minimum requirement to obtain unambiguous depth information. With only one image, there is the intrinsic ambiguity of the 3D to 2D mapping, and could lead to some perspective illusions. By means of triangulation, we can infer the depth of most points in the stereo images, if we are able to find corresponding points in the two images, or the disparity map.

The first step of depth reconstruction is image rectification, which is a well-solved problem, and is in the course material; we can obtain more accurately rectified image pairs using stereo cameras. The next step is estimating the disparity map, *i.e*. estimating the distance between correspondent points. The last step is triangulation; the principle of triangulation is a similarity triangle problem, see Fig 1. We can calculate the depth Z of a point P, if we know its disparity $d = x_R - x_T$ in the two images, using the length of base line b and the camera focal length f:

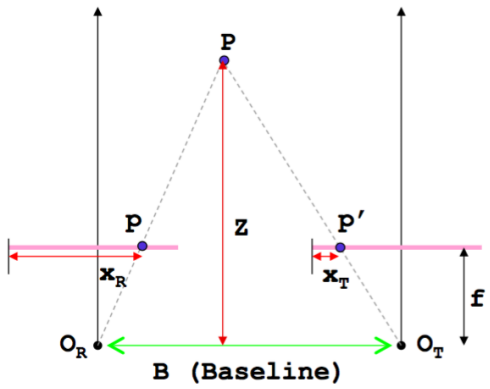


Figure 1. Triangulation. Computing depth from disparity.

$$Z = \frac{b \cdot f}{d} \tag{1}$$

So the depth reconstruction problem can be simplified as estimating the disparity map from a stereo image pair.

The problem can be stated as: given a rectified stereoscopic image pair (a reference image R and a target image T), where the y coordinates of each corresponding points pair are the same, we need to obtain a uni-valued *disparity map* $d(x, y)$. The (x, y) coordinates of the disparity map are taken to be coincident with the pixel coordinate of the *reference image*. The correspondence between a pixel (x, y) in the reference image R and a pixel (x', y') in the target image T is then given by

$$x - d(x, y) = x', y' = y. \tag{2}$$

Usually, a conservative bound on the maximum disparity value $disp\_max$ is given or can be estimated.

In this course project, we first adopt the sequence alignment algorithm to compute disparity, the initial results are not very satisfying, though through hand-tuning parameters

and adding various algorithms into the pipeline, the performance may improve. But our reasoning is that the sequence alignment is prone to error and may be worse than other disparity computation algorithms.

We switch to other methods. The state-of-the-art algorithm is [2] (at that time it ranks first on both KITTI and KITTI2015 leaderboards[1], but now it ranks second on KITTI2015), which utilizes a very large dataset to learn the disparity information from end to end, without hand-tuning many parameters and stacking many algorithms, though we still need to tune the hyper-parameters when training. And it outperforms many mature pipelines, *e.g.*, [9]. So we reimplement this algorithm to see whether it is that powerful.

2. Related work

Binocular stereo matching, the procedure of obtaining disparities, is an old problem in computer vision, and has been heavily investigated. [7] generalized stereo matching algorithms to four components: matching cost computation, cost (support) aggregation, disparity computation / optimization, and disparity refinement. [8] made a classification and evaluation specifically for the cost aggregation component. [4] introduced many stereo matching algorithms in the history. With the development of deep learning, Zbontar and LeCun [9] first proposed to train a convolutional neural network (MC-CNN) on pairs of small image patches where the true disparity is known. The output of the network is used to initialize the matching cost, before that, the matching cost is computed from the raw image pixels. MC-CNN improves the performance by a large amount, and it is now commonly be used in the stereo matching pipeline. For example, the method [3] ranked first on the Middlebury leaderboard[2].

In the big data era, Mayer *et al*. [6] proposed three large synthetic stereo datasets, and presented a convolutional network for real-time disparity estimation. More deep approaches are proposed. Recently, Kendall *et al*. [2] proposed an end-to-end deep learning architecture for regressing disparity, and its performance is stated in Sec. 1.

3. Approach

3.1. Adopting sequence alignment algorithm

Adopting sequence alignment algorithm to do the disparity computation is possible, so our first approach is as follows: using the four components [7] to refine the results step by step, adopting MC-CNN (we use the MC-CNN-arct model, its architecture is shown in Fig 2 to compute the initial matching cost, applying some cost aggregation

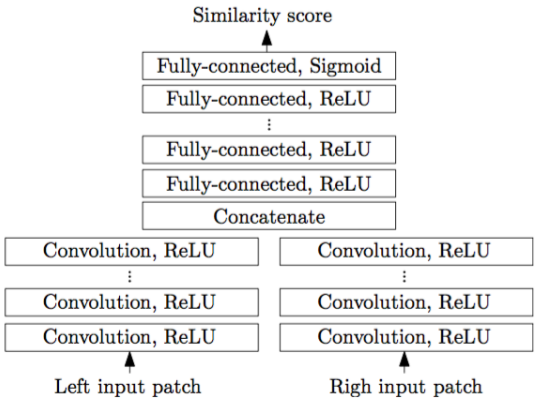


Figure 2. The MC-CNN-arct architecture [9].

algorithm (we use [10] in our experiments), using the sequence alignment algorithm to compute disparity, adopting some disparity refinement algorithms (common methods are left and right consistency check, interpolation, subpixel enhacement *etc*., usually need more than one refinement algorithm). The reasoning for using [10] is: since sequence alignment algorithm only takes a single horizontal line into consideration, we need to use some techniques to make the similarity score of each pixel include the information of its vertical neighbors.

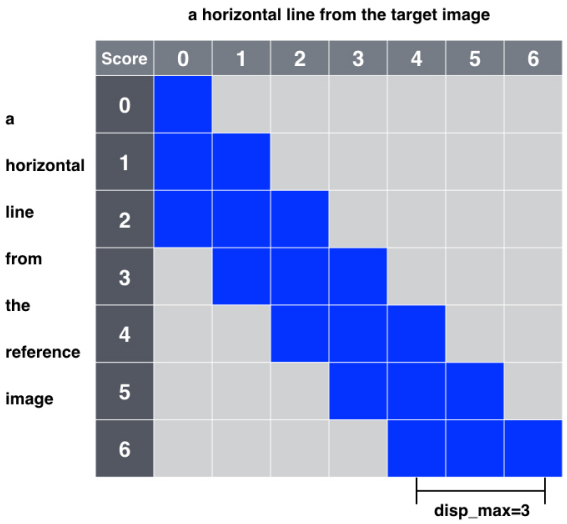


Figure 3. Due to the constraints on the possible disparity, the memory usage is greatly reduced. The figure is a tiny example, $W = 8$ and $disp\_max = 3$. The adapted algorithm only takes up the memory space of the blue squares.

we adopt the Needleman-Wunsch algorithm[3] to do the sequence alignment. As for how we adapt the sequence

[1] http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo, http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo

[2] http://vision.middlebury.edu/stereo/eval3/

[3] https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm

alignment algorithm, which is originally work on genome or text sequences, to compute the disparity: the algorithm is performed on each horizontal line of the images. The insertion and deletion operation in this algorithm corresponds to skipping a pixel in the target and reference images respectively, substitution corresponds to the matching of two points. By finding the best path receiving the maximum score, we find all the matching points pairs in the whole horizontal line.

The memory usage of this dynamic programming algorithm is greatly reduced due to the constraints on the disparity. Suppose the reference image is always the image taken by the left camera, then for the matching points pair, the coordinates must satisfy $x_R \geq x_T$ and $x_R - x_T < disp\_max$. Suppose the image width is W, instead of using $O(W \times W)$ memory space, we only need to use $O(W \times disp\_max)$, which is illustrated in Fig 3.

The tricky part lies in the score definition of the three operations. We use the following score scheme in the experiments (larger score means better option. The matching cost obtained from MC-CNN ranges from 0 to 1, the smaller the more similar):

$$\begin{aligned} &Del(x) = 0.1 \\ &Ins(y) = 0.1 \\ &Sub(x,y) = \begin{cases} 0, & \text{if } match(x,y) > 0.5 \\ 1 - match(x,y), & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

3.2. End-to-end Learning of Geometry and Context

The large-scale synthesized dataset SceneFlow [6] makes training deep neural network for regression possible. Using Middlebury or KITTI, the datasets' scales are too small for regression DNN training.

[2] is an interesting end-to-end method, which reduces the trial and error of hand-tuning many parameters, and of fitting various algorithms to the four components. With a well-defined deep neural network architecture, we only need to tune the hyper-parameter and train the model. And it outperforms all the complicated traditional methods on the two KITTI leaderboards. The whole network architecture is illustrated in Fig 4 and Fig 5 (image copied from the paper).

Compared with the previous work, its major difference is the cost volume, which corresponds to the $(max\_depth + 1) \times height \times width$ cost volume in the geometry of stereo vision. Instead of simply concatenating the left and right feature maps, GC-Net concatenating each unary feature with their corresponding unary from the opposite stereo image across each disparity level, and packing these into a 4D volume.

After computing cost volume, it utilizes 3D convolution to refine the cost volume, and uses the deep encoder-and-decoder technique to reduce the computational complexity.

GC-Net also adopts a differentiable argmin function, which is the sum of the all the depth, weighted by their softmax confidence.

Since the paper stated most of the details clearly, we reimplement this paper using TensorFlow (following the paper), and conducts experiments on two benchmark datasets.

4. Experiments

4.1. Initial results of using sequence alignment

We conduct the initial experiments on the first training image pair in the KITTI 2012 stereo dataset.

Firstly, we use the MC-CNN-arct model [9] to obtain the initial 3D matching cost volume with shape $[disp\_max, H, W]$. That is, for every point (x, y) in the reference image, we estimate the cost of matching it with the point $(x - d, y)$ for each d in the range $[0, disp\_max)$. Using the simple argmin function, we obtain a very rough disparity estimation. Applying the sequence alignment algorithm in Sec 3.1 to this cost volume, we obtain a preliminary result.

Since the matching cost obtained from the output of the neural network is computed on 9×9 patch pairs, which is rather small considering the image size (370×1226), we adopt the cross-based cost aggregation algorithm [10] to aggregate the matching cost, and then apply the sequence alignment algorithm to the aggregated matching cost volume.

The qualitative results for each step and the input image pair are shown in Fig 6.

Though the qualitative results look fine, the quantitative results (the number of pixels whose error is larger than K and the average error) is not satisfying. The most likely reason is that the score scheme is not good. We increase the constant score of deletion and insertion to 0.2, but there is no improvement. And compared with the whole pipeline used in [9], this pipeline's performance is much worse.

The score scheme is hand-tuned, and cannot be perfect. Under the sequence alignment algorithm, since the score used to determine the stereo matching is accumulated through the whole horizontal line, the error introduced by the imperfect score scheme and other noise also accumulates. So we decided to switch to other methods, to reimplement the GC-Net [2].

4.2. Datasets

We use two datasets in the following experiments: the two KITTI stereo datasets [1, 5] (combined as one dataset), and SceneFlow [6].

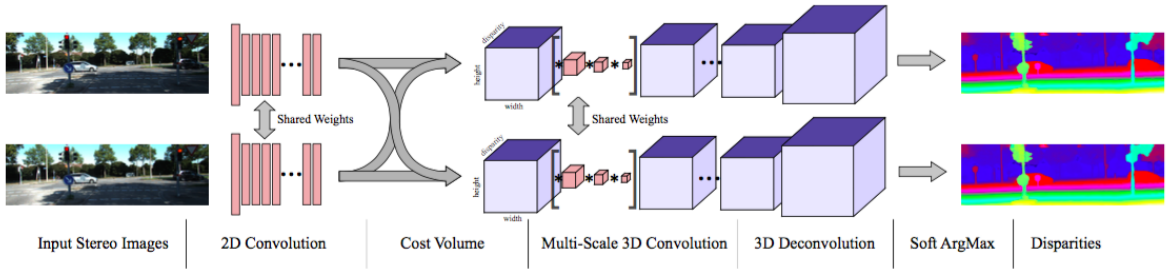


Figure 4. The deep stereo regression architecture, GC-Net [2].

	Layer Description	Output Tensor Dim.
	Input image	H×W×C
	Unary features (section 3.1)	
1	5×5 conv, 32 features, stride 2	½H×½W×F
2	3×3 conv, 32 features	½H×½W×F
3	3×3 conv, 32 features	½H×½W×F
	add layer 1 and 3 features (residual connection)	½H×½W×F
4-17	(repeat layers 2,3 and residual connection) × 7	½H×½W×F
18	3×3 conv, 32 features, (no ReLu or BN)	½H×½W×F
	Cost volume (section 3.2)	
	Cost Volume	½D×½H×½W×2F
	Learning regularization (section 3.3)	
19	3-D conv, 3×3×3, 32 features	½D×½H×½W×F
20	3-D conv, 3×3×3, 32 features	½D×½H×½W×F
21	From 18: 3-D conv, 3×3×3, 64 features, stride 2	¼D×¼H×¼W×2F
22	3-D conv, 3×3×3, 64 features	¼D×¼H×¼W×2F
23	3-D conv, 3×3×3, 64 features	¼D×¼H×¼W×2F
24	From 21: 3-D conv, 3×3×3, 64 features, stride 2	⅛D×⅛H×⅛W×2F
25	3-D conv, 3×3×3, 64 features	⅛D×⅛H×⅛W×2F
26	3-D conv, 3×3×3, 64 features	⅛D×⅛H×⅛W×2F
27	From 24: 3-D conv, 3×3×3, 64 features, stride 2	1/16D×1/16H×1/16W×2F
28	3-D conv, 3×3×3, 64 features	1/16D×1/16H×1/16W×2F
29	3-D conv, 3×3×3, 64 features	1/16D×1/16H×1/16W×2F
30	From 27: 3-D conv, 3×3×3, 128 features, stride 2	1/32D×1/32H×1/32W×4F
31	3-D conv, 3×3×3, 128 features	1/32D×1/32H×1/32W×4F
32	3-D conv, 3×3×3, 128 features	1/32D×1/32H×1/32W×4F
33	3×3×3, 3-D transposed conv, 64 features, stride 2	1/16D×1/16H×1/16W×2F
	add layer 33 and 29 features (residual connection)	1/16D×1/16H×1/16W×2F
34	3×3×3, 3-D transposed conv, 64 features, stride 2	⅛D×⅛H×⅛W×2F
	add layer 34 and 26 features (residual connection)	⅛D×⅛H×⅛W×2F
35	3×3×3, 3-D transposed conv, 64 features, stride 2	¼D×¼H×¼W×2F
	add layer 35 and 23 features (residual connection)	¼D×¼H×¼W×2F
36	3×3×3, 3-D transposed conv, 32 features, stride 2	½D×½H×½W×F
	add layer 36 and 20 features (residual connection)	½D×½H×½W×F
37	3×3×3, 3-D trans conv, 1 feature (no ReLu or BN)	D×H×W×1
	Soft argmin (section 3.4)	
	Soft argmin	H×W

Figure 5. Details of the GC-Net architecture.

The two KITTI datasets are collections of rectified image pairs taken from two video cameras mounted on the roof of a car, and a rotating laser scanner mounted behind the left camera recorded ground truth depth, labeling 30% of the image pixels. The ground truth are noisier. The 2012 dataset contains 194 raining and 195 testing image pairs, and the 2015 dataset contains 200 training and 200 testing images. For the training image pairs, the ground truth disparity maps and the masks indicating the points with valid ground-truth disparity are given. The KITTI datasets have online leaderboards for evaluation, where people can submit the test results of their methods to compare with the state of the art. So the ground-truth disparity of the testing pairs is withheld.

SceneFlow is a large-scale synthetic dataset, consisting of three subsets (each subset using different synthetic techniques). We combine the three subsets and divide the whole dataset into two subsets: 35,981 training and 843 testing image pairs. For the FlyingThings3D subset, we use the predivided training and testing subset, for the other two subset, we use the ratio of $5:1$. Since it is very large, we use a batch size of 1, and the whole training process will cover only several epochs, so it is very hard to overfit, and we do not use a validation set.

For SceneFlow, when training, we use a 256×512 randomly located crop from the input images. When testing, we use a 256×512 center located crop from the input images. For KITTI, we split the input images into four evenly-sized images to fit the GPU memory.

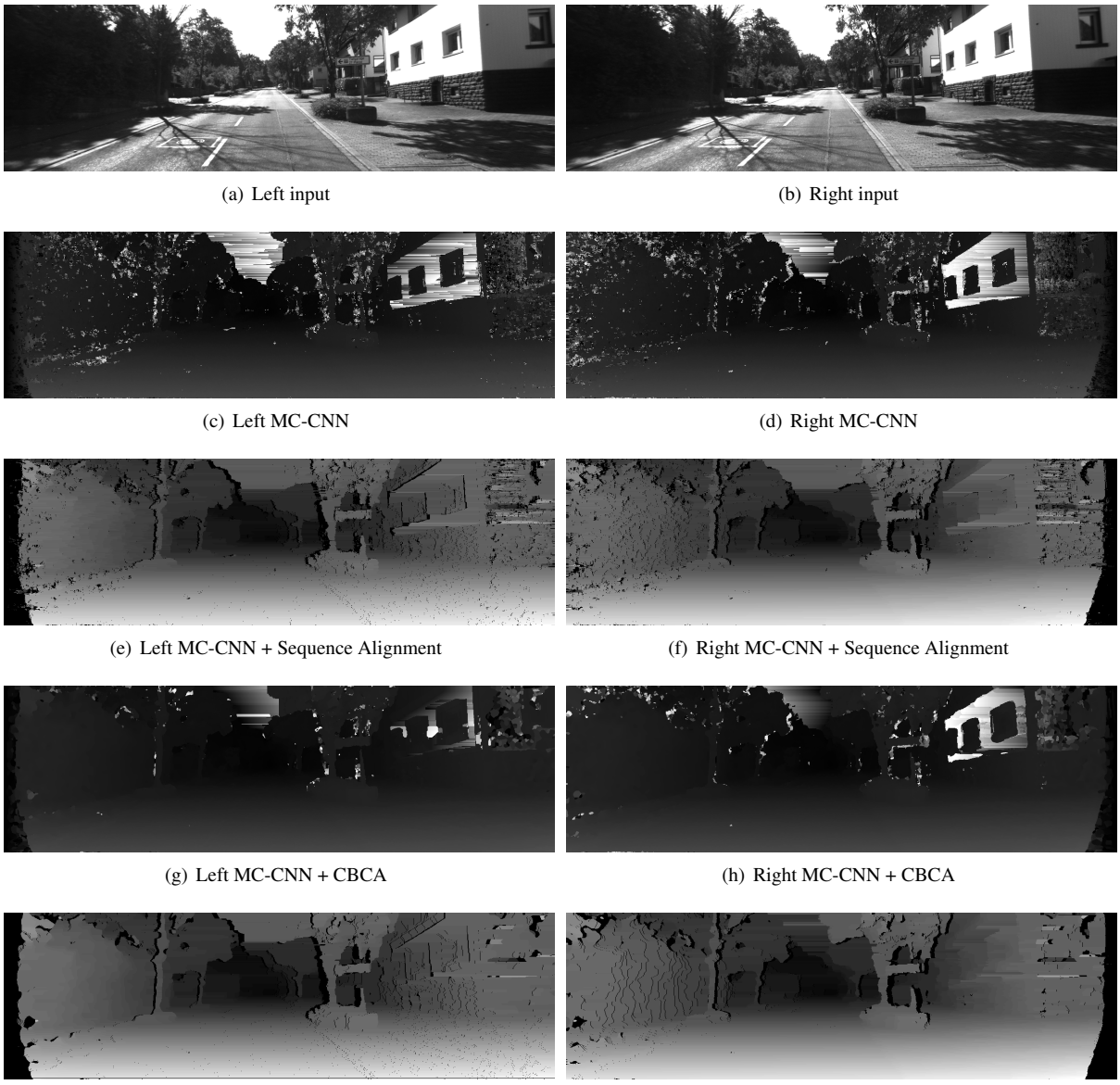
4.3. Implementation details

Follow the paper, we use the TensorFlow Framework, and implement the model using all the built-in `tf.nn` functions. We follow every details written in the paper. The only difference is that I implement a multiple-GPU version to speed up the training. But there are some details missing from the paper, for the variable initializers, we follow the Resnet implementation in the TensorFlow model zoo. For the hyper parameters of the RMSProp optimizer, we follow the Inception implementation at first, since the training runs not very well, we vary the hyper parameters later. We also decrease the learning rate to 0.0001 after 40K iterations training. Our experiments run on Titan-X GPUs.

4.4. Results of GC-Net

Following the paper, We first train from scratch on SceneFlow, for about 50K iterations (with batch size 4, run 1 pair of images on each GPU), and then finetune on KITTI, for about 10K iterations (with batch size 3). The number of iterations we use is adjusted based on the training results and the loss curves.

Since the KITTI dataset is sparse, with a disparity mask,



(a) Left input (b) Right input

(c) Left MC-CNN (d) Right MC-CNN

(e) Left MC-CNN + Sequence Alignment (f) Right MC-CNN + Sequence Alignment

(g) Left MC-CNN + CBCA (h) Right MC-CNN + CBCA

(i) Left MC-CNN + CBCA + Sequence Alignment (j) Right MC-CNN + CBCA + Sequence Alignment

Figure 6. Sequence alignment qualitative results

so we train the GC-Net model with a disparity mask, to mask out the inconsistent and occluded points. The SceneFlow dataset does not provide such masks, after communicating with the dataset's authors, we run left and right consistency checks on the left and right ground truth disparity maps to form the mask, using the formula in [9]. However, this causes some problem in the training: visualizing the predicted results, we find that the model tends to predict blurry and averaged disparity maps. It may because we use a too small threshold in the consistency check, and in this way make too much masks. But if the model is very good, it should learn to regress disparity from the unmasked parts (we average the loss on the unmasked points, so the loss is normalized). Besides, for datasets like KITTI, the provided training ground truth is with a mask, we still think the methods should work with a mask, maybe this will require a better designed network.

Dataset	MAE	Bad@3	Bad@5	Bad@7
SceneFlow test	14.04	0.66	0.53	0.45
KITTI train	2.86	0.28	0.146	0.0866

Table 1. Results when training with masks.

For both datasets, we use the common evaluation criteria $Bad@K$: the percentage of pixels where the true disparity

and the predicted disparity differ by more than K pixels. Usually, only errors on non-occluded pixels are counted. We will also present some qualitative results.

The quantitative results are shown in Table 1. We do not have qualitative results on KITTI test set (remember we do not have the ground truth of the test set). And due to some both resource and time limitations, we do not do validation. Since the current model has some space for improvement, and one can only submit limited times on the leaderboard, I do not submit it to the leaderboard. The qualitative results on SceneFlow are shown in 7, subfigure a to d. The qualitative results on KITTI are shown in 8, including samples from both training set and testing set. Though the test results on SceneFlow is bad, but the results on KITTI is not bad. The qualitative test results seem fine. Maybe because KITTI's ground truth is blurry and sparse, after applying the mask, the predictions seem good.

After training with the masks, during testing, we find that if using fixed mean and variance (calculated in the training process) in the batch normalization layer, the results are bad, see Fig 7, subfigure e and f. However, if using the real-time mean and variance, the results are better, especially on the KITTI dataset.

To try to obtain better results, we remove the masks, and train the model from scratch. This time, the visualized results seem better, and is shown in Fig 7, subfigure g and h. But time is limited, we cannot obtain the new qualitative results before deadline, so we represent some temporary qualitative results (The model is still training, we show the training samples prediction results).

5. Conclusion

Through my exploring on the two stereo matching approaches, one traditional, one using deep neural network. The traditional one requires much trials and errors, and it is hard to outperform the deep approaches (according to the leaderboards, the methods ranked higher are all CNN heavy). However, the deep approach, based on my experience, is very tricky, and requires great deep neural networking tuning techniques. The model architectures are correct, why the results (currently, the training results of the training without mask version seem still cannot reproduce the high performance of the original paper) are so different is beyond me.

References

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

[2] 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. *arXiv preprint arXiv:1703.04309*, 2017.

[3] L. Li, X. Yu, S. Zhang, X. Zhao, and L. Zhang. 3d cost aggregation with multiple minimum spanning trees for stereo matching. *Applied Optics*, 56(12):3411–3420, 2017.

[4] S. Mattoccia. Stereo vision algorithms for 3d dense reconstruction. University of Florence, 2012.

[5] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] 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.

[7] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

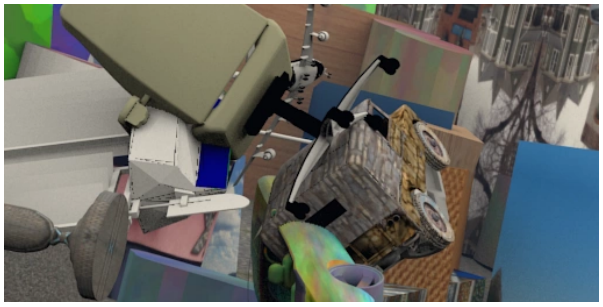
[8] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda. Classification and evaluation of cost aggregation methods for stereo correspondence. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[9] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32, 2016.

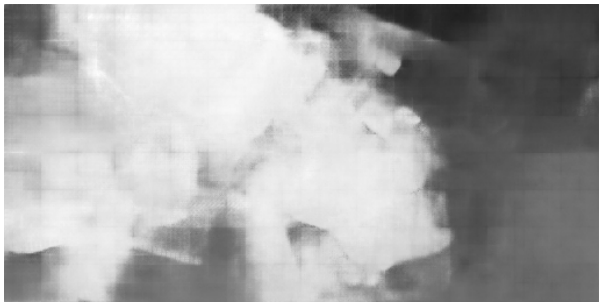
[10] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(7):1073–1079, 2009.

(a) Train with mask, groundtruth

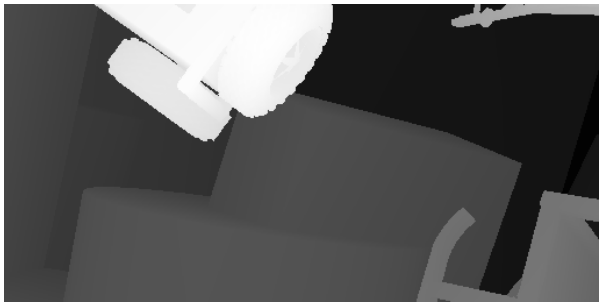
(b) Train with mask, masked prediction



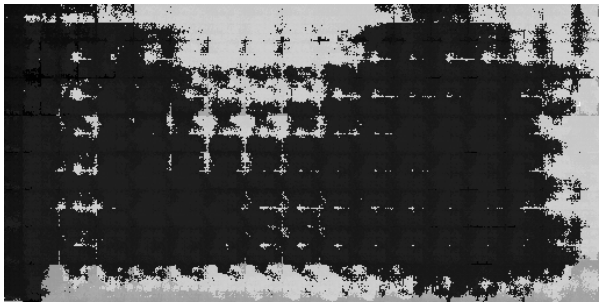
(c) Train with mask, original image



(d) Train with mask, prediction



(e) Train with mask, bad when using fixed mean/var, groundtruth

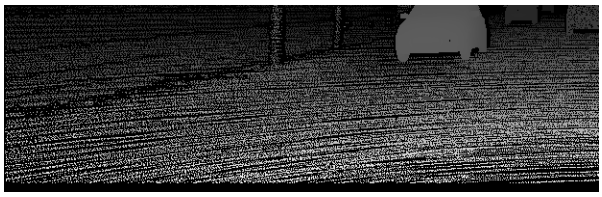


(f) Train with mask, bad when using fixed mean/var, prediction

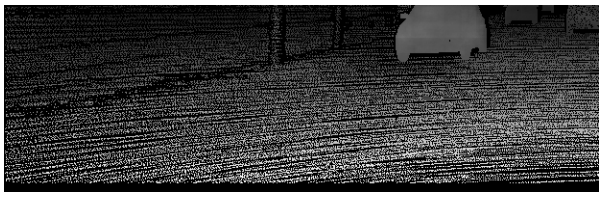
(g) SceneFlow train without mask original image

(h) SceneFlow train without mask prediction

Figure 7. GC-Net qualitative results on SceneFLow



(a) Perform on training set, ground truth 1



(b) Perform on training set, masked prediction 1

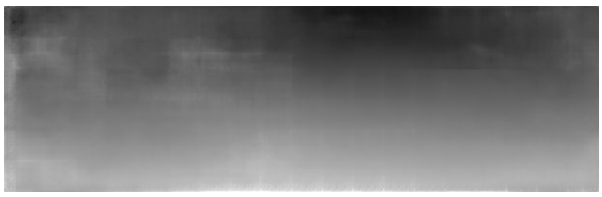
(c) Perform on training set, ground truth 2

(d) Perform on training set, masked prediction 2

(e) Perform on test set, original image 1

(f) perform on test set, prediction 1

(g) Perform on test set, original image 2



(h) Perform on test set, prediction 2

Figure 8. GC-Net qualitative results on KITTI