Convolutional network for stereo matching and comparison with traditional methods

Minchuan Zhou
Stanford University
mczhou@stanford.edu

Abstract

Convolutional networks and traditional stereo matching algorithms are both state-of-the-art methods, which can sometimes be combined to achieve more accurate results. We implement a Siamese network that uses the inner product of the output of two branches as the matching cost, and show their favorable performance, compared to traditional algorithms with a matching cost of absolute difference. Besides this first step of match cost computation, we also implement a second step of cost aggregation with average pooling and a third step of disparity optimization with semi-global matching, which further improves the accuracy of the computed stereo map. We also show that by using dilated convolution in the network architecture, we can achieve comparable accuracy while greatly improving the computational efficiency.

1. Introduction

Stereo vision is one of the most important research topics in computer vision, with its application in robotics, autonomous driving and augmented reality. Stereo vision is the extraction of 3D information such as shapes and appearance from 2D images. It has been one of the most important topics in the area of computer vision. A important area of stereo vision is stereo matching, which is to extract the depth information from a pair of rectified images taken from left and right cameras (or eyes). Here we want to produce the disparity map, encoding the disparity value, which is the difference in location of the object in the image of left and right cameras. Assuming a focal length f for the cameras, a camera separation of B, and a disparity of d, we can calculate the depth of the object from:

$$D = Bf/d \tag{1.1}$$

Traditional stereo matching algorithms measure the similarity using the matching cost, usually absolute difference (AD) or squared difference (SD). More advanced traditional methods are also being developed [1,2]. Like other traditional machine learning algorithms, traditional stereo matching algorithms usually requires manual choice of some parameters. As the rapid development of deep learning techniques like convolutional neural network and computer hardware, many new stereo matching algorithms are developed which can achieve much better accuracy than traditional methods. In the meanwhile, traditional methods are still playing important roles in many aspects.

In this report, we implement a state-of-art deep learning stereo matching algorithm, based on a Siamese network architecture, and compare its performance with traditional method based on a matching cost of absolute difference. We evaluate both the computational efficiency and accuracy quantified by the three pixel error. For further refinement, we also utilize average filtering for cost aggregation and semi-global matching for disparity optimization. The link to the git repository is https://github.com/harrainy18/cs231a_project.git.

2. Related work

2.1. Traditional stereo matching algorithm

Traditional stereo matching algorithms includes both local and global approaches [3]. The local approach uses sliding window, utilizing a simple matching cost of absolute difference (AD), squared difference (SD), normalized cross correlation (NCC), etc. These algorithms may fail for situations such as repeated patterns and textureless regions. An example of global methods is the energy minimization method [4], which add smoothness to the cost function, based on the assumption that adjacent pixels should move about the same amount. In general, local approaches runs faster but gives less accuracy than global approaches.

There are generally four steps in a stereo matching algorithm — matching cost computation, cost aggregation, disparity computation/optimization, and disparity refinement [3]. The first and most important step is matching cost computation. The matching cost can be computed using AD, SD, NCC, etc. Cost aggregation can be done using average filtering or cross based cost aggregation. There are many ways of

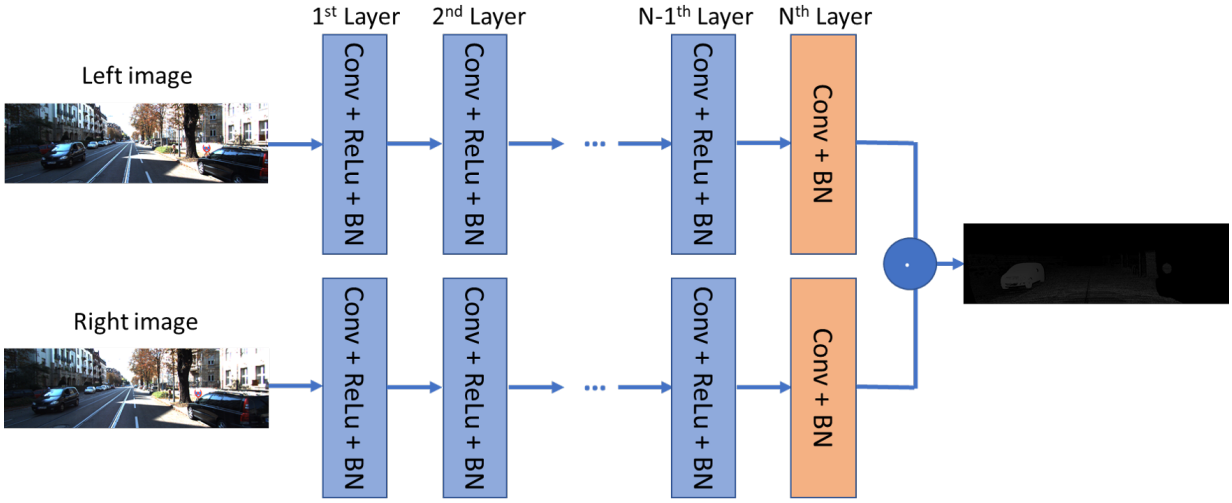


Figure 1. Architectures of CNN for stereo matching. The number of layers N is 9 for Net1 and 5 for Net2.

disparity optimization, such as graphic cut, scanline optimization and semi-global matching [5]. Finally, the disparity map can be refined to deal with problems such as invalid matches occlusion. The methods for disparity refinement include slated plane smoothing and left-to-right-consistency check.

2.2. Deep learning for stereo matching

CNN advances the development of many aspects of computer vision, such as image recognition, video analysis, image segmentation, 3D reconstruction, etc. Many CNN based stereo matching algorithms are developed [6, 7, 8]. Zbontar and LeCun [**6**] Implemented the MC-CNN network to compute the matching cost for disparity map. The left and right images are each sent into a CNN, and the outputs are are concatenated and sent to more layers for further processing. W. Luo et al. [**8**] proposed a different Siamese network architecture. In the model, a Siamese network is used to extract marginal distributions over all possible disparities for each pixel and an inner product layer to connect the two branches of the network. Using a simple product operation instead of additional CNN layers, Luo et al. was able to achieve much faster computation. The matching cost is computed from the Siamese network, then cost aggregation is applied by average pooling and further optimized using semi global block matching (SGM) or/ and slanted plane approach.

3. Approach

3.1. Traditional method

The difference of various stereo matching algorithms generally lies in how the matching cost is computed. The matching cost is a function of the pixel location in the image (x_i, y_i) and disparity $C(x_i, y_i, d)$, where $d \in (-d_{\max}, d_{\max})$. Here $d_{\max}$ is the maximum disparity and $d_{range} = 2d_{\max} + 1$ is the range of all possible disparities. We use a simple matching cost of AD in our implementation. We have also tested SD, which gives similar results to AD. After cost computation, the disparity map is computed using a "winner takes all" approach:

$$d_{est}(x_i, y_i) = \underset{d}{\operatorname{argmin}}\ C(x_i, y_i, d) = \underset{d}{\operatorname{argmin}}\ |I^L(x_i, y_i) - I^R(x_i + d, y_i)| \tag{3.1}$$

3.2. CNN for stereo matching

We implement the siamese network architecture in Ref. **8**. The architecture of the network, which we name as "Net1", is shown in Figure 1, with 9 layers ($N = 9$). The left and right images are sent to the same convolutional network, respectively. Then an inner product is taken for the output from the two networks to get the cost volume $C(x_i, y_i, d; \Theta)$, where Θ denotes the parameters of the model.

We use a total of 9 layers in each CNN network. The kernel size is chosen to be 5×5 with a stride of 1. The first three layers we use 32 filters, and for the rest we use 64 filters for each layer. Batch normalization is used in each layer to help with the issue of internal covariate shift that slows down the training process.

Rectified Linear Units (ReLu) are used for all the layers except for the last layer. We use valid padding in each layer and the size of the patch is chosen to be the same as the receptive field so that in the final output, the size of the patch is 1.

For training the network, we divide the left images into small patches of size $n_{patch} \times n_{patch}$. Assume the center of the patch is located at (x_i, y_i), and the ground truth disparity value is $d(x_i, y_i)$. Since the images are rectified, the epipolar lines are horizontal so we just need to search the matched patch along the horizontal direction. So we can extract a patch of size $n_{patch} \times (n_{patch} + d_{range})$ centered at the pixel $(x_i + d(x_i, y_i), y_i)$ in the right image.

The intuitive choice for the loss function that quantifies the difference between the probability distribution from the network output and a target distribution that peaks at the ground truth disparity value is a cross-entropy loss function. We use the cross-entropy loss function with a target distribution:

$$L = -\sum_i p_{target}(d_i) \log p(d_i; \Theta) \qquad (3.2)$$

where we apply a softmax to the cost function to get $p(d_i; \Theta)$, and the target distribution is

$$p_{target}(d_i) = \begin{cases} 0.5 & \text{if } |d_i - d_{gt}| = 0 \\ 0.2 & \text{if } |d_i - d_{gt}| = 1 \\ 0.05 & \text{if } |d_i - d_{gt}| = 2 \\ 0 & \text{otherwise} \end{cases} \qquad (3.3)$$

The target distribution we use is a discrete distribution. Naturally, we can choose a smooth distribution, for example a Laplacian distribution or a Gaussian distribution. Since Laplacian distribution has a less heavier tail, we use Laplacian in the training.:

$$p_{target}(d_i) = \frac{1}{2b} \exp\left(-\frac{|d - d_{gt}|}{b} \right) \qquad (3.4)$$

where we choose $b = 1$. We will also test if using such a target distribution could improve the performance of training.

To decrease the computational cost, we use another network architecture that makes use of dilated convolution. The architecture of the second network (Net2) is the same as that in Figure 1, with only 5 layers ($N = 5$). We use dilated convolution in the network [9] to increase the receptive field. With only 5 layers, we achieve the same receptive field as the original 9 layer network.

The network is trained using using stochastic gradient descent back propagation with Adam instead of AdaGrad in the original paper. AdaGrad decays the learning rate aggressively because of the growing denominator, while Adam solves the problem by decaying the denominator as well. The learning rate is set to 0.01 first, and then after 24k iterations it is decreased by a factor of 5 for every 8k iterations [3].

3.3. Cost aggregation and disparity optimization

Cost aggregation and disparity optimization steps can be used after the cost computation step to improve the disparity estimation.

The idea is based on the assumption that the depth and therefore disparity value changes smoothly. We perform cost aggregation using average pooling, in order to reduce the noise.

We implement the disparity optimization step using semi-global matching (SGM). We define an energy function $E(D)$ of the disparity map D, adding smoothness terms that penalize the discontinuity in disparity between adjacent pixels [**6**].

$$\begin{aligned} E(D) = \sum_p \Bigg(& C(p, D(p)) \\ & + \sum_{q \in N_q} P_1 \{ |D(p) - D(q)| = 1 \} \\ & + \sum_{q \in N_q} P_2 \{ |D(p) - D(q)| > 1 \} \Bigg) \end{aligned} \qquad (3.5)$$

where $p = (x, y)$, and N_q represents neighbors of the pixel p. The penalty for a disparity change of 1 is P_1 and that for a disparity larger than 2 is P_2.

Specifically, we use the matching cost for each direction r to minimize the energy $E(D)$:

$$\begin{aligned} C_r(p, d) = C(p, d) - \min_k C_r(p - r, k) \\ + \min \Big\{ & C_r(p - r, d), C_r(p - r, d - 1) + P_1 \\ & + C_r(p - r, d + 1) + P_1, C_r(p - r, k) + P_2 \Big\} \end{aligned} \qquad (3.6)$$

In our implementation of SGM, we use only 4 directions of r to perform line optimization. Typically, 8 directions or 16 directions are used for better quality but with slower computation.

Except for the constant penalty used in Ref. **6**, we can also use a linear penalty term,

$$P = \tau |D(p) - D(q)| \qquad (3.7)$$

which increases with the magnitude of disparity

change. We evaluate both the constant and linear SGM in our experiment.

4. Experiments

4.1. Dataset for training and evaluation

We use the KITTI 2015 data set [10], which includes a collection of image pairs (200 training pairs and 200 testing pairs) taken from two video cameras mounted on the roof of a car. The image pairs are rectified and the ground truth disparity maps measured using a rotating laser scanner. In our model, we use a patch size of $n_{patch} = 37$, and therefore obtain about 12 million patches for training from 160 training images. The rest 40 images are used for evaluation of the model. For training purpose, the patch images are grouped into mini-batches of 128 patches and sent to the network. To get a better accuracy, we would like to use the scene flow dataset [11], which has more than 39,000 stereo frames in 960x540 pixel resolution. But due to limited computational resources, it is not possible for now.

For evaluation of the implemented stereo matching algorithms, we will follow the 3px error in KITTI 2015 data set: the percentage of bad pixels averaged over all ground truth pixels whose disparity error is larger than 3px. We compute the 3px error for the 40 validation images. As noted in the KITTI 2015 dataset, a value of 0 indicates an invalid pixel, so we exclude pixels with 0 value in the evaluation.

4.2. Training time and testing time

A summary of training and testing time is shown in Table 1. The traditional method does not require training, while the deep learning models need to be trained. Since the architecture Net2 has fewer layers than Net1, the training time of Net2 is half of the training time for Net1, reduced from 450 ms/step to 220 ms/step.

In testing, the traditional method of matching cost computation takes 3 times more time than Net1 or Net2. Most of time is taken in SGM used for disparity optimization, which takes 100 more time than traditional cost volume computation. In our implementation, we use 4 directions of line optimization. Using 8 directions increase the computational time by a factor of 6, due to the much slower speed for diagonal directions.

4.3. Convergence in network training

The loss function decreases with the number of iterations. As can be seem from Figure 2, the loss function starts to converge around 20k iterations, for both two CNN architectures Net1 and Net2. The convergence curve is quite fuzzy. Increasing the size of the sample for training would help with this issue. Throughout this report, we use 50k iterations for training.

We also investigate using a smooth Laplacian

Table 1. Training and testing time. The testing time for Net1, Net2 and traditional methods does not include the time spent on average filling and SGM.

	Training time (ms/step)	Testing time (sec)
Net1	450	1.7
Net2	220	1.6
Traditional	—	4.7
Average filtering	—	0.5
SGM	—	468

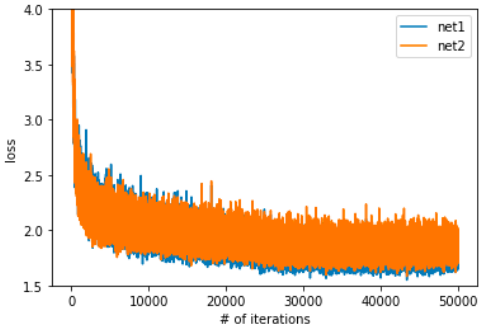


Figure 2. Plot of loss as a function of number of iterations in training Net1 and Net2.

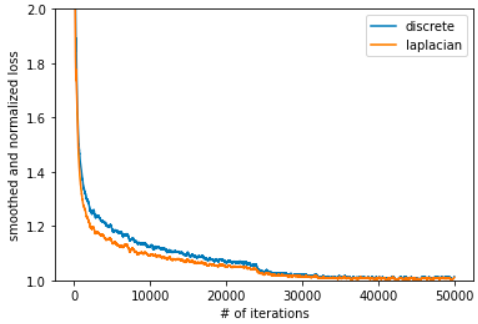


Figure 3. Plot of smoothed and normalized loss as a function of number of iterations in training Net1 with the discrete and Laplacian target distribution.

distribution as the target distribution. Figure 3 shows a comparison of the loss function, which is smoothed and normalized to the minimum value. It can be seen that the loss function using a Laplacian target distribution converges slightly faster than the case with a discrete target distribution.

4.4.Accuracy of disparity map estimation

We first evaluate our traditional method. An example of the disparity map estimation is shown in Figure 4. The disparity map calculated directly using a matching cost of AD is shown in Figure 4 (d), with a high 3px error of 50%. The error is reduced to 6.5% after cost aggregation using average pooling with a filter size of 5×5. The disparity map (Figure 4 (e)) much less noisier, but there is still some discontinuity in the disparity map. Since SGM adds smoothness to the disparity, the disparity discontinuity can be reduced by SGM using constant penalty, which is indeed observed in Figure 4 (f). With SGM, the error is reduced to 3.7%. Using SGM with linear penalty, we get a similar error of 3.6%. We can see that without cost aggregation and disparity optimization, the raw disparity map is very noisy and of low accuracy.

Evaluating the performance on real-world data of KITTI2015 validation set, we find that without further processing the disparity estimation has very low accuracy, with an average 3 px error of 81% over the 40 evaluation images. With average filtering with a filter size of 7×7, the disparity map less noisier than that before cost aggregation but still with a high error of 39%. After constant SGM optimization, the average 3 px error is 25%; while using linear SGM, we can get the error down to 12%. The results are summarized in **Table 2**.

Table 2. 3 pixel error of disparity estimation for different methods.

3px error	Raw	After average filtering	After constant SGM	After linear SGM
Traditional	81%	39%	25%	12%
Net1	5.9%	5.6%	4.7%	4.3%
Net2	6.0%	5.7%	4.9%	4.4%
Net1 (Laplacian)	5.9%	5.6%	4.8%	4.6%

In Figure 5 (c), we show an example of disparity map estimated using the traditional method. The disparity map is very noisy, with a 3 px error of 80%. In addition, the algorithm fails for textureless regions, as can be seen from the disparity value for the road. It also fails for the shadow on the walls, where there is a jump in the luminance of the image. We also notice that for reflective surfaces, the accuracy of disparity estimation is poor, for example, for the car in front. The disparity estimation is also difficult for regions

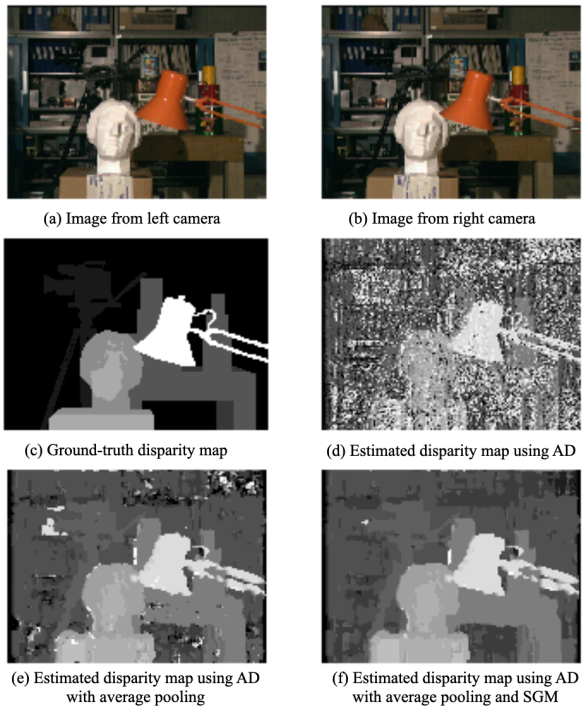


Figure 4. An example of disparity estimation for Tsukuba scene.

with jumps in disparity (e.g., the leaves on the trees) and occlusions (e.g., the row of cars on the right).

The disparity map after cost aggregation is shown in Figure 5 (d), with an error of 48%. The noise in the disparity map is reduced but the problems we mention above still exist. Then we use SGM with constant penalty to further optimize the disparity, as shown in Figure 5 (e), with 36% error. The accuracy of textureless regions and the shadow is improved. SGM with linear penalty can greatly reduce the error to 23%, as shown in Figure 5 (f). Compared to the result of constant SGM, the disparity estimation for the regions with jumps in disparity is improved.

For deep learning stereo matching, we have trained the networks Net1 and Net2 as described in Figure 1. For Net1, the average 3 px error is 5.9%. The deep learning method, even without cost aggregation and SGM, has much better accuracy than the traditional method. With average filtering with a filter size of 5×5, the 3 px error is reduced slightly to 5.6%. When constant or linear SGM is applied, we get a 3 px error of 4.7% or 4.3%, respectively.

An example of the estimated disparity map with Net1 is shown in Figure 5 (g), showing 21% error without CA and SGM. The region with reflective surfaces and occlusions, for example the cars, shows some big error. The disparity map after applying average filtering is shown in Figure 5 (h), with an error of 20%. We find that average pooling indeed clears out some noise in the disparity map but the accuracy of disparity estimation does not improve much. We further apply a linear SGM in Figure 5 (i), which shows an error of 14%. The discontinuity in the disparity map represented by the random variations are removed. We can see that the result is more accurate than the result of traditional method in Figure 5 (f), especially in the region with reflective surfaces.

The example we discuss above is a difficult case for disparity estimation. The easiest case in the dataset is shown in Figure 6. Here, there is no difficulties caused by occlusions or jumps in disparity. The result of Net1 with average filtering and SGM shows a 3px error of 2.02%, while the traditional method shows an error of 2.5%. Thus, for a simple case, the traditional method can achieve a comparable accuracy with the deep learning method.

The other deep learning model (Net2) with dilation and fewer layers gives a comparable 3 px error of 5.7% when cost aggregation is applied, and an error of 4.4% with cost aggregation and SGM with linear penalty. The accuracy is comparable to Net1, while the training time is reduced by a factor of 2. An example of the

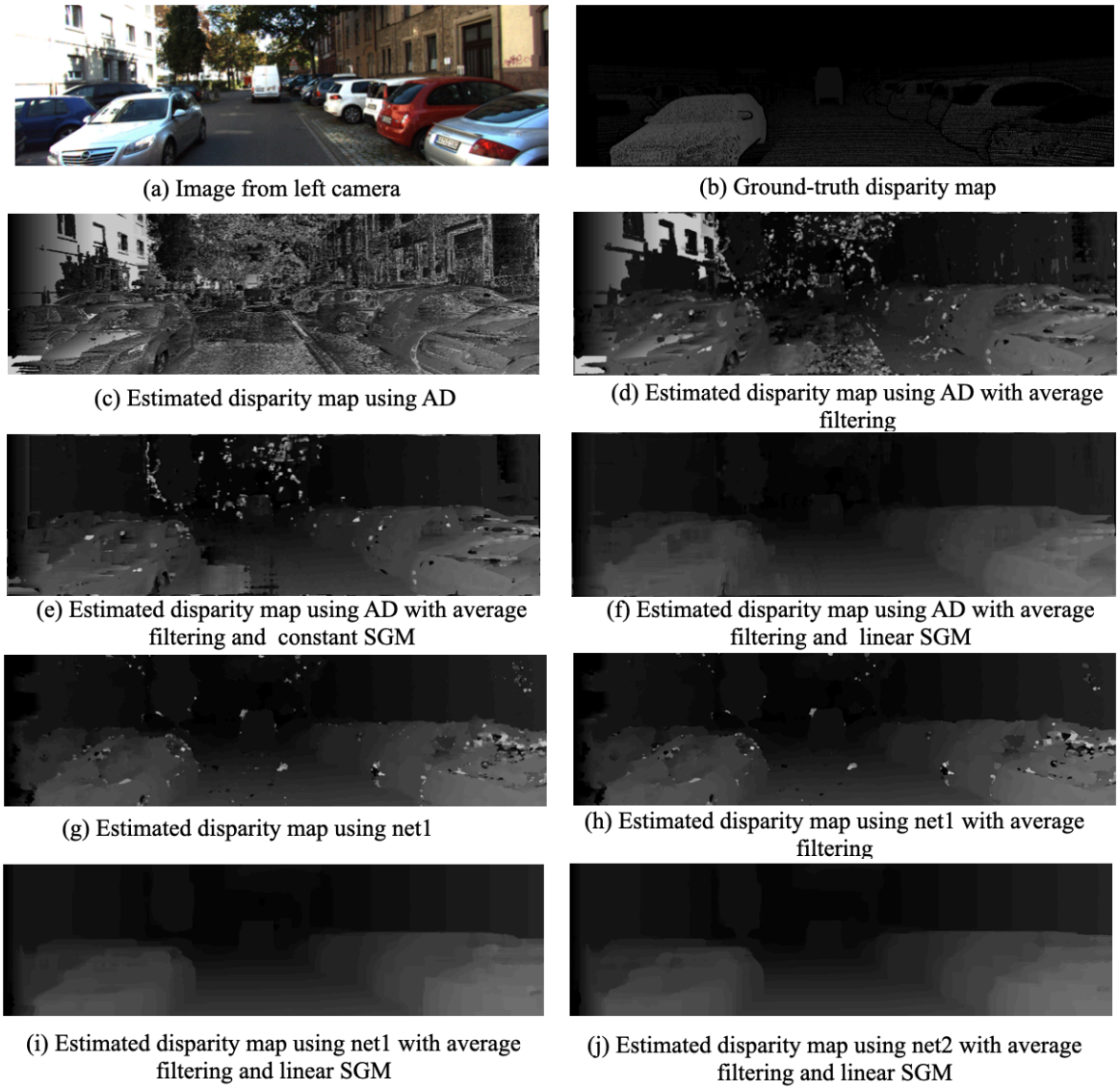


Figure 5. Examples of ground-truth and estimated disparity maps from KITTI2015 dataset.

disparity map is shown in Figure 5 (j). Comparing figures (h) and (j), we can see that the results is similar for the two different CNN architectures.

When we use a smooth Laplacian target distribution instead of the discrete distribution for training the network Net1, the average 3 px error is 5.6% with average filtering, which is almost the same as the case of discrete target distribution. The error using linear SGM is 4.6%, slightly larger than 4.3% for discrete target distribution.

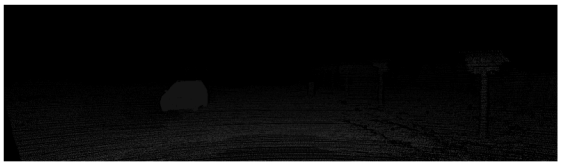
5. Discussion and future work

We have implemented two stereo matching algorithms: a deep learning method and a traditional method with a matching cost of absolute difference. We have shown that the deep learning method of matching cost computation is more computational efficient for testing than the traditional method, but requires pre-training, unlike traditional methods. The deep learning method has better accuracy than the traditional method, especially for the difficult situations, for example, pictures with occlusions and reflective surfaces. For simple cases, traditional method can provide comparable accuracy. We have implemented the cost aggregation step using average filtering. We have also implemented the disparity optimization step using semi-global matching, which enforces smoothness constraint to the disparity map.

Two different structures of networks (Net1 and Net2) were trained and tested. The architecture Net2 makes use of dilated convolution to reduce the computational cost without compromising the performance of the network. We also find that the convergence with a smooth target distribution is slightly faster than that with a discrete target distribution.

It can be seen that that the disparity map may still contain errors due to occlusions or invalid matches, even after cost aggregation and disparity optimization. For future work, we would like to implement the disparity refinement step to get more accuracy estimation. We notice that the most time consuming part in our implementation is the SGM step for cost optimization. In the future, we will modify our implementation to improve the computational efficiency of SGM. To improve the network training, we would like to use a larger dataset for training.

(a) Image from left camera



(b) Ground-truth disparity map

(c) Estimated disparity map using traditional method with average filtering and SGM

(d) Estimated disparity map using net1 with average filtering and SGM

Figure 6. An example from KITTI2015 dataset that is an easy case for disparity estimation.

References

1. Bai C, Ma Q, Hao P, Liu Z, Zhang J. Improving stereo matching algorithm with adaptive cross-scale cost aggregation. *International Journal of Advanced Robotic Systems*. January 2018. doi:10.1177/1729881417751544.
2. Sangeetha, G.R., Kumar, N., Hari, P.R., Sasikumar, S., 2018. Implementation of aStereo vision based system for visual feedback control of Robotic Arm for spacemanipulations. Proc. Comput. Sci. 133, 1066–1073.https://doi.org/10.1016/j.procs.2018.07.031.
3. Hamzah, R.A., Ibrahim, H.,. Literature survey on stereo vision disparity map algorithms. J. Sensors 2016. https://doi.org/10.1155/2016/8742920
4. Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, November 2001, pp. 1222-1239, vol. 23, doi:10.1109/34.969114
5. Hirschmuller H. Stereo processing by semiglobal matching and mutual information[J]. IEEE Transactions on pattern analysis and machine intelligence, 2008, 30(2): 328-341.
6. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015.
7. Chen, J., Yuan, C., 2016. Convolutional neural network using multi-scale information for stereo

matching cost computation. In: Proc. – Int. Conf. Image Process. ICIP 2016-Augus, 3424–3428. https://doi.org/10.1109/ICIP.2016.7532995.
8. W. Luo, A. G. Schwing and R. Urtasun, Efficient Deep Learning for Stereo Matching, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 5695-5703, doi: 10.1109/CVPR.2016.614.
9. T. Wang, M. Sun and K. Hu, Dilated Deep Residual Network for Image Denoising, 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, 2017, pp. 1272-1279.
10. A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite, *CVPR*, 2012.
11. https://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html