Computer Vision Final Project
Depth Generation on More Realistic Scenes

B04901096 蔡昕宇

Introduction

I chose deep stereo methods at first to implement this task due to the difficulty of synthetic images. However, I did not get a better result on both synthetic data and real data. Due to the domain difference between synthetic and real ones, deep methods somehow have some difficulties in learning in multiple domains based on limited modality of the training data. In other words, the model somehow cannot act well on both synthetic data and real data. Hence, traditional methods seem to be more feasible on this task. Details will be introduced later on, included the deep methods I implemented.

Algorithms

The algorithm is similar to the work I did in the HW4. It is referred to the paper "*On Building an Accurate Stereo Matching System on Graphics Hardware*", which computes the matching cost from ADCensus cost (absolute difference cost (C_{AD}) + census cost (C_{CENSUS})). Most parts are identical to the work I've done in Hw4. However, there are some new challenges in these testing data, explained as follows, and I implemented some different new techniques that I have done in this task.

Preprocessing

The major difference between the work in HW4 is in the preprocess, since the maximum disparity is not given and the image is not quite standard. The following is my implementation of the matching process.

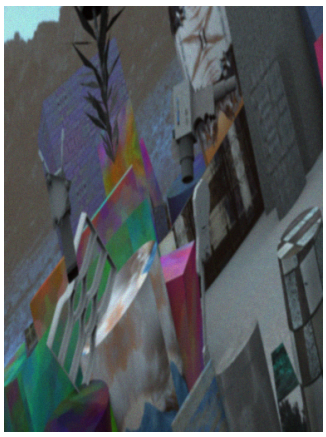
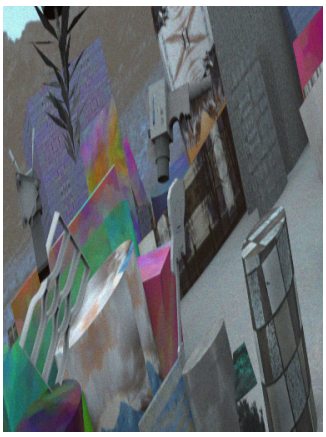
Disparity defining

In order to find the possible disparity range without just take a look at the answer which is impossible in the real data, I used the feature matching process such as SIFT, SURF[1]. Some unique patterns in both left and right image can be find in the matching process. After finding some good matches between images, we can calculate the distance between each matching pairs. Hence, this maximum disparity can be set by the distance just calculated. I found that there are some distance are negative, especially in real data. However, I did not handle the negative distance and just consider the positive ones, and this also somehow reduced my computation time.

Color Difference Between Pairs

It is is that the data may have some difference in its lightness or brightness. Take the example pair of the synthetic data, TL0.png and TR0.png.

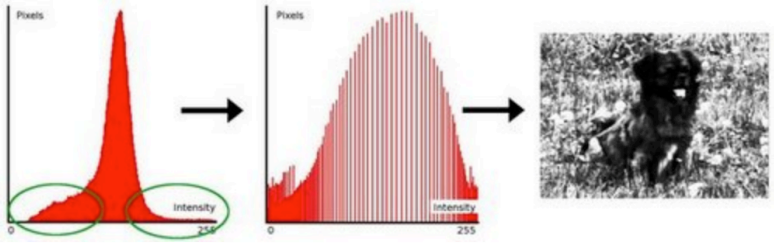
[1] Reference: Feature Matching, https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html

Take a closer look in both images. The left one is TL0.png and the the right one is TR0.png. It is obvious that the right image is more brighter than the left one. It is a severe problem in this algorithm, affecting C_{AD} and C_{CENSUS}. The absolute difference of the RGB space can be distorted due to the luminous problem of the paired images. Hence, I done some preprocess for the input images.

Histogram Equalization

Histogram equalization aims to smooth the color intensity of an image, mapping the distribution of the color intensity to the another distribution. The color intensity will be smooth over the range of 0 to 255. This allowed us to smooth the intensity difference of our testing data, since we have significant difference in our data.



This is a sample[2] of a distribution mapping by means of this method. Take the example shown above, we can reduce our intensity difference as shown below.

We then can apply our algorithm after this process, since the color intensity does not differ greatly.

[2] Ref: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html

Bilateral Filter

It is possible that it exists great difference on the same surface in subtle areas. Hence, bilateral filtering can reduced the high frequency noise generated by those slanted or rugged surface, and keep the edges clearly without ambiguity. I will later show the difference between the results with and without bilateral filtering.

The following process is almost the same to HW4.

Cost Computation

Absolute Difference Cost + Census Cost (ADCensus)

C_{AD} measures the color difference between the windows scanned in the left image and the right image. Besides, C_{CENSUS} encodes local information with relations between colors of each pixel into binary strings. Therefore, this work considers both costs which considers patterns and intensity simultaneously.

Cost Aggregation

Cross-based Cost Aggregation

In this paper work, they proposed a new strategy is to aggregate costs horizontally and vertically in alternative turns. This strategy are able to maintain color consistency in both horizontally and vertically, not just one of them.

The 3 constraints (rule) constructed for extending the arm of the window is that (take the left arm as example)

1. $D_c(p_l, p) < \tau_1$ and $D_c(p_l, p_l + (1,0)) < \tau_1$
2. $D_s(p_l, p_l) < L_1$
3. $D_c(p_l, p) < \tau_2$ if $L_2 < D_s(p_l, p) < L_1$

Rule 1 restricts the color difference between p_l and p, and it restricts p_l and its predecessor so that the arm will not run across the edges in the image. Rule 2 simply constrained the maximum length. Rule 3 is a more strict constraint when p_l is far from p, since according to the setting, $\tau_2 < \tau_1$. The arm extends only when the color patterns are close enough.

Disparity optimization

Scanline Optimization

Scanline optimization performs in four directions: up-down, down-up, left-right, right-left. It will update the cost for each pixel by the formula

$$C_r(p,d) = C_1(p,d) + min(C_r(p-r,d)), C_r(p-r, d \pm 1 + P_1, \min_k C_r(p-r,k) + P_2) - \min_k C_r(p-r,k)$$

Intuitively, it compares the cost with the predecessor of each pixel, and the parameters P_1, P_2 penalize the disparity changes between neighboring pixels. The value of P_1 or P_2 depends on the color difference between query pixel and its predecessor both

in left image and right image. After that, I averaged the cost computed in four directions, obtaining the results after the optimizing process.

Disparity Refinement

Outlier detection

First, for the refinement process, we need to find out which pixels we need to improve from its previous prediction. A pixel p is an outlier if it does not satisfy the rule

$$D_L(p) = D_R(p - (D_L(p),0))$$

Intuitively, it tells us that p in the left image is an outlier if it does not match the disparity calculated from the view of the right image. Hence, we have to calculate another disparity map from right image. This process also finds the mismatch point and occlusion point. During the matching process, if there is no intersection of its epipolar line and D_R, it is labelled as an occlusion point, which can not be seen in both images at once. This process doubles the computation time, but it reduces the error largely.

Iterative Region Voting

The second step is to fill in this outliers with other disparity value. In order to fill in this outliers, we should find its neighbors to find the most reliable one to fill in this outlier pixel. Take advantage of the window computed in the aggregation process, we find the reliable one in this window through histogram. We find those matched points and collect their disparity into histogram. It helps us to find the disparity with the highest probability intros region. It replaces the original disparity if it has enough amount of reliable information. That is, to satisfy the following inequality:

$$S_p > \tau_S, \frac{Hist(d_p^*)}{S_p} > \tau_H$$

Differ from HW4, the process does not clear the outlier at the edge of image. It somehow depends on the threshold τ_H, and also on the cost volume calculated from previous process. Hence, here we should modify the threshold due to the bad quality of cost computation.

Proper Interpolation

The remaining outliers are filled be interpolation. We now find the optimal point in 16 directions. For outlier p, if p is an occlusion point, we find the pixel with the lowest disparity value since p probably comes from the background. Otherwise, for the mismatch points, we find the pixel with the most similar color. Then we replace p from those selected pixel.

Depth Discontinuity Adjustment

After processing the outliers, we should consider about the edges in the image. We find all the edges by sobel filter[3]. For the pixel is on the edge in the image, we take both

[3] Ref: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.sobel.html

p_1 and p_2 from both sides of the edge, and pick the one with lower matching cost. Finally, we replace p with the pixel which has lower cost.

Sub-pixel Enhancement

Sub-pixel enhancement can efficiently reduce the errors cause by discrete discontinuity disparities. In this step, using a quadratic polynomial interpolation form as

$$d^* = d - \frac{C(p, d+1) - C(p, d-1)}{2(C(p, d+1) + C(p, d-1) - 2C(p, d))}$$

to interpolate three candidates d, $d+1$, $d-1$. This process greatly smoothes the edges of the disparity map, which later can be seen in the results.

Median Filter

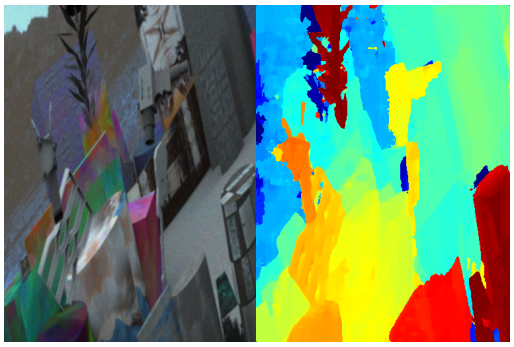
I used a 3 x 3 median filter[4] which smoothes the interpolating results and the edges and get our final results.

Results

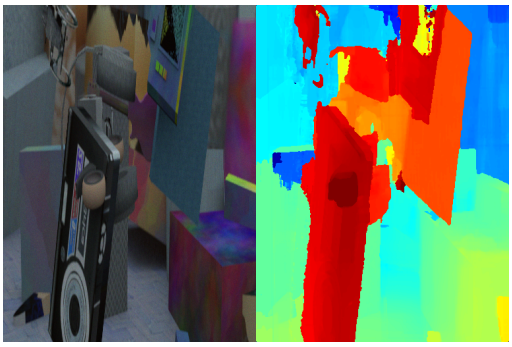
I found that some results will be good without C_{AD} and bilateral filter. We will later discuss the effectiveness of C_{AD} and the bilateral filter.

Synthetic data

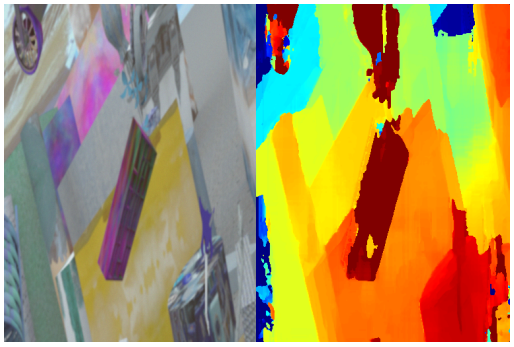
TL0.png



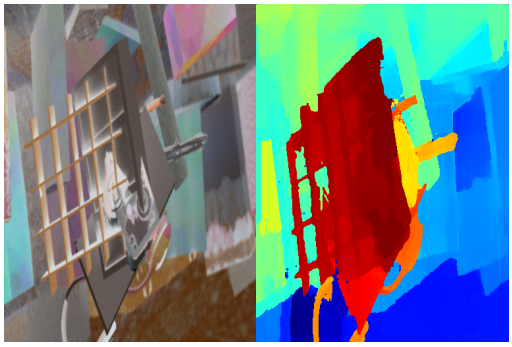
TL1.png



TL2.png

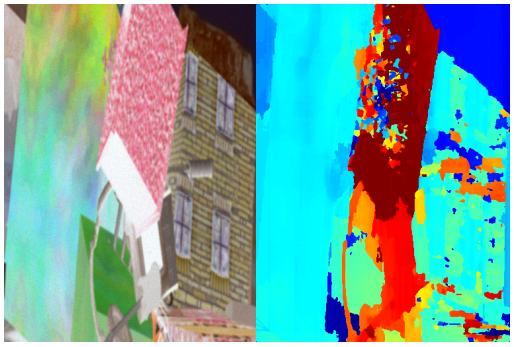


TL3.png

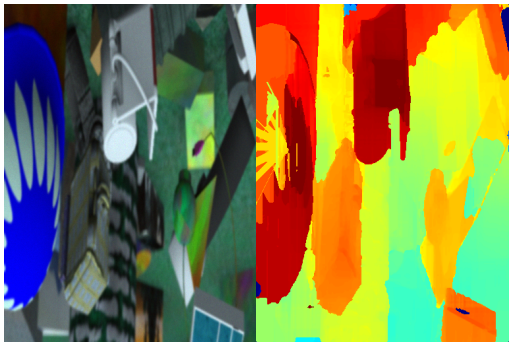


[4] Ref: https://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html

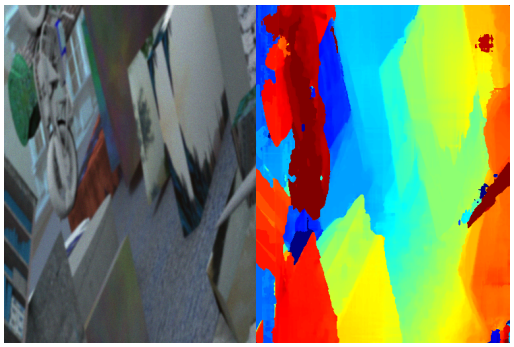
TL4.png



TL5.png

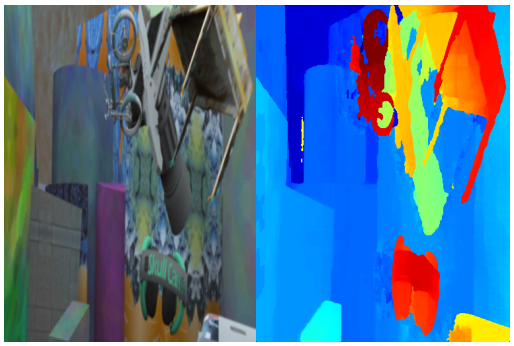


TL6.png

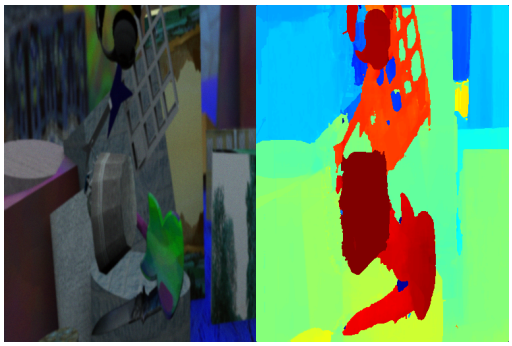


TL7.png

TL8.png

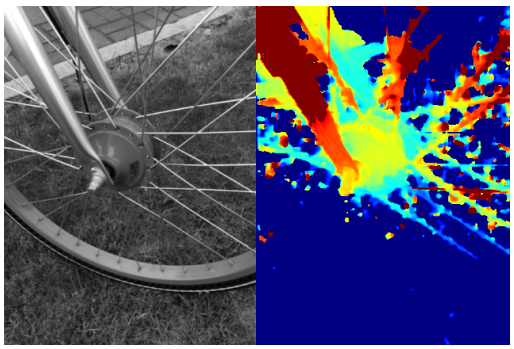


TL9.png

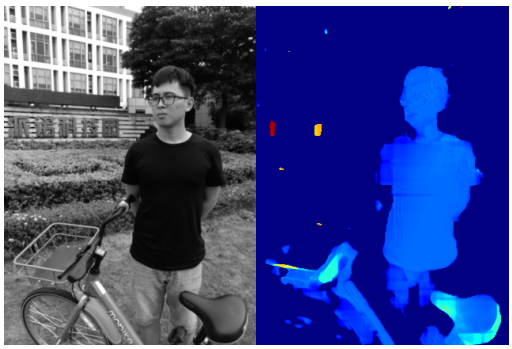


Real data

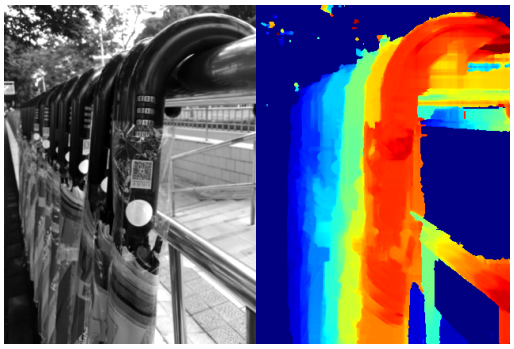
TL0.png



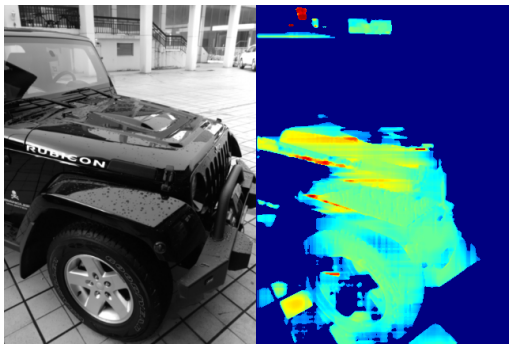
TL1.png



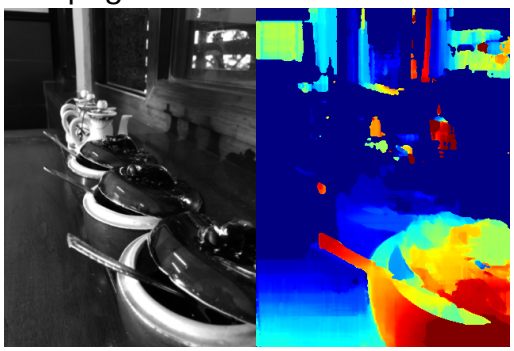
TL2.png



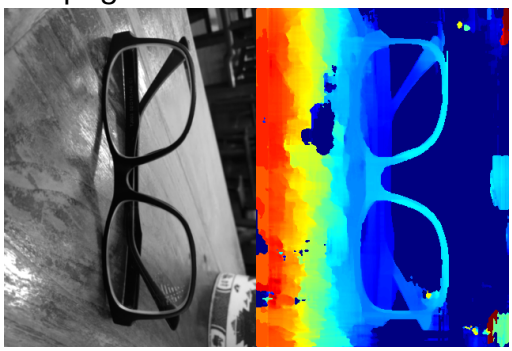
TL3.png



TL4.png

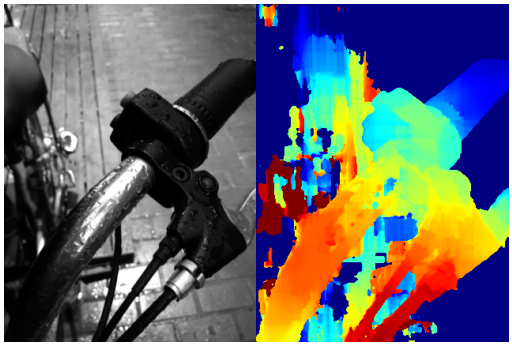


TL5.png

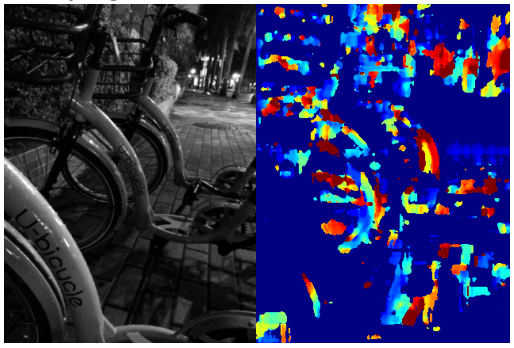


TL6.png

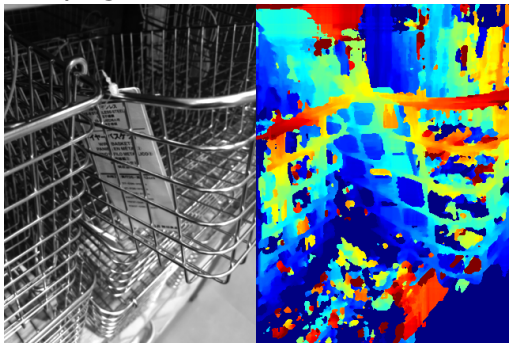
TL7.png



TL8.png



TL9.png



Error of Synthetic Data

TL0.png	TL1.png	TL2.png	TL3.png	TL4.png	**Average Error**
1.854	1.700	4.543	2.777	4.789	
TL5.png	TL6.png	TL7.png	TL8.png	TL9.png	2.713
3.083	1.881	3.379	1.358	1.766	

Discussion

Problems in Images

There are four important issues in the paired stereo images: occlusion and texture-less surfaces, photometric variations, repetitive patterns, and reflection.

The occlusion problem is solved by the left-right consistency, while the repetitive patterns are solved by scanline optimization. These are the issues that are already solved in the paper. Photometric problems is a severe problem just mentioned in the beginning. I solved it with histogram equalization, which smoothed the photometric and intensity difference between images. I didn't handle the texture-less edges. Take TL3.png in the real data, the window of the car and the background is too similar so that my algorithm cannot detect the edge. I think that I can improve it by some segmentation methods, allowing us to get access to more information of the edges. Maybe I will may fix this problem in the future. Last but not the least, reflection is an important issue that makes my disparity map having extreme disparity values, just as the result in TL8.png of the real data. TL8.png of real data has large amount of negative disparity, which I did not handle. Therefore, I get a bad result in TL8.png.

Computation time

Machine Spec: Intel Core i5, 2.7GHz

Synthetic (larger disparity)	Real (smaller disparity)
about 4500s / per image	about 1500s / per image

Effectiveness of Bilateral Filter

The error without bilateral filter is shown in the chart as below

TL0.png	TL1.png	TL2.png	TL3.png	TL4.png	**Average Error**
2.399	1.919	5.058	3.009	7.204	
TL5.png	TL.png	TL7.png	TL8.png	TL9.png	3.492
3.753	2.504	4.330	1.536	3.211	

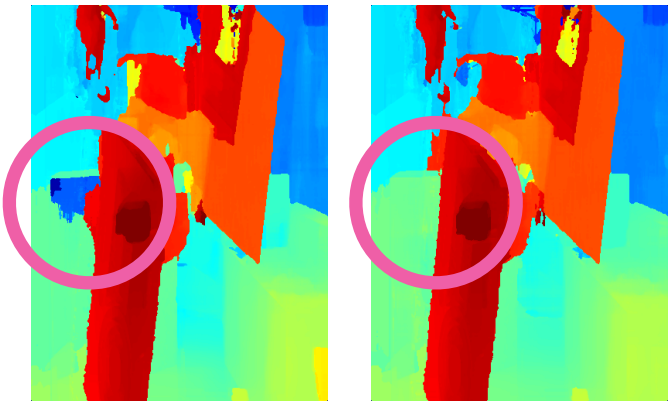
The result is relatively higher than that of the process with bilateral filter. Hence, I implemented the bilateral filter in my final result.

Effectiveness of Absolute Difference Cost

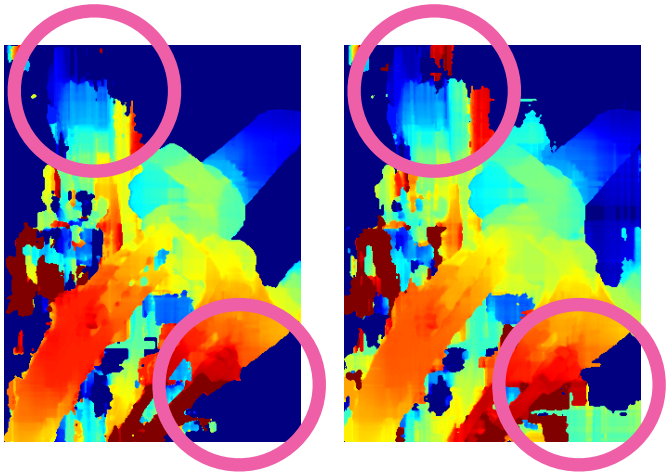
The error without C_{AD} is shown in the chart as below

TL0.png	TL1.png	TL2.png	TL3.png	TL4.png	**Average Error**
1.898	1.458	5.177	2.828	4.338	
TL5.png	TL6.png	TL7.png	TL8.png	TL9.png	2.616
1.964	1.912	3.348	1.419	1.815	

The result is somehow relatively lower than that of the process with calculation of C_{AD}. There are some results are relatively better than that of the final results, but the others worse than the final results. For example, some parts can be dealt better without C_{AD}, as the two figures shown below (left: with C_{AD}, right: without C_{AD})



Clearly, some areas can be smoothed by removing C_{AD}. In some real data cases do not perform as well as synthetic ones. Take the example (TL7.png) as the following images:



Removing the C_{AD} cost causes the algorithm can not distinguish the foreground and background pattern, like the example in real data. Therefore, it may somehow generate better outcome of synthetic data. Overall, it generates lower average loss but does not gives good result after removing absolute difference cost.

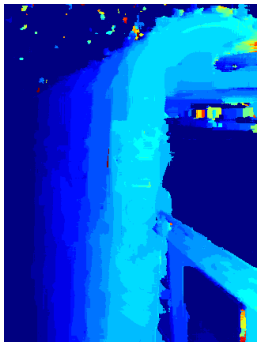
Deep Algorithms Discussion

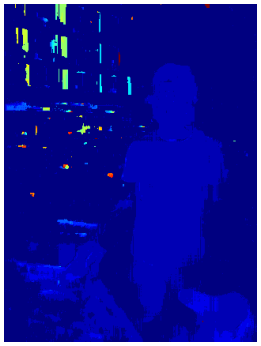
I implemented some deep models to learn the disparity map. I used the extra dataset KITTI2015[5]. Hence, I got a good result in real data, but I cannot fit the synthetic data. Learning based model are limited by learning source during its training process. Therefore, we need a cross domain learning process. However, state-of-the-art works rarely done on this dataset with cross-domain learning. Maybe this is a field that should be further discovered.

[5] KITTI2015 dataset: http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo

I implemented the work in 2016 called "Efficient Deep Learning for Stereo Matching"[6]. Recently, more and more end-to-end methods came into present, and also predicts good disparity map. Some state-of-the-art methods such as PSMNet[7], GCNet[8] also perform well on real image dataset. Back to the paper I implemented, the model structure is similar to the structure of MC-CNN[9], which learns the relation between randomly sampled image window patches. The work directly minimize the 3-pix loss during the training process. After training, I take the cost volume trained during the process as the cost initialization, and then going through the optimization and refinement process just as same as that in traditional methods.

I did not put those 10 real data in training, but it predicts a results which is similar to the result from my traditional methods. The following are some sample outputs of my learning model.



However, I failed to fit on the synthetic data, where the shapes generated in the images seems not to be like any object in the origin data. I can't find out the bugs in the training code. Hence, I return to the traditional methods as mentioned above.

[6] W. Luo et al., Efficient Deep Learning for Stereo Matching, *CVPR 2016*

[7] J. R. Cheng, Y. S. Chen, Pyramid Stereo Matching Network, *CVPR 2018*

[8] A. Kendall et al., End-to-End Learning of Geometry and Context for Deep Stereo Regression, *CVPR 2017*

[9] J Žbontar, Y. LeCun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, *JMLR 2016*