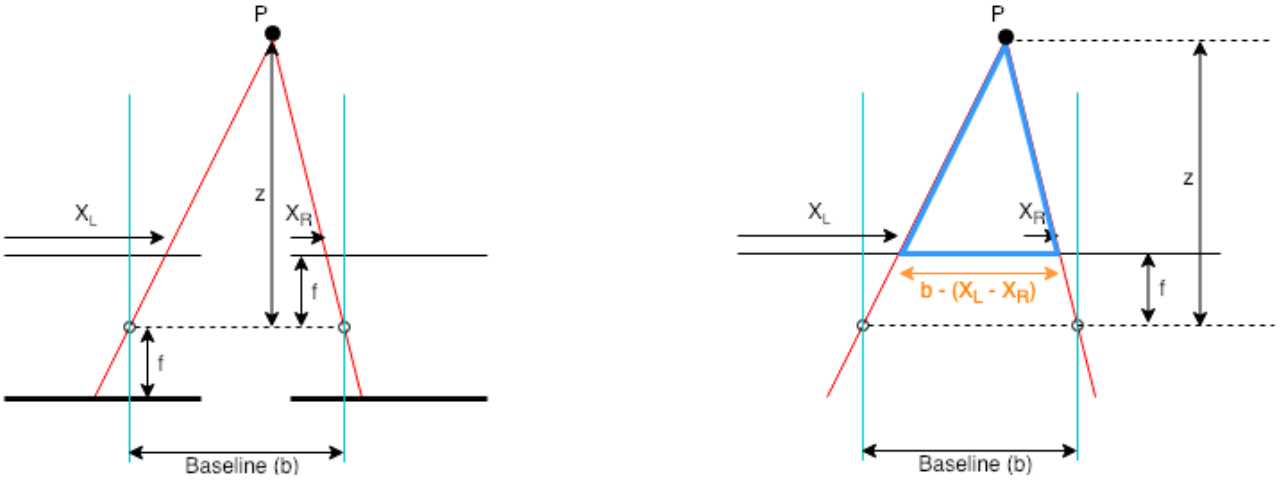
Computer Vision Hw4 Stereo Matching

B04901096 蔡昕宇

Part 1 Depth from Disparity

I rearranged the representation of the distance X_L and X_R as the left-bottom figure to



make it more easier to form an equation.

After that, we can know that the length highlighted in color orange is (b - X_L - X_R), since the distance between two paired images is equal to the length of the baseline. By the formula of similar triangles, we can obtain that

$$\frac{b-(X_L-X_R)}{b}=\frac{b-d}{b}=\frac{z-f}{z}$$

Rearrange the formula, we can get the final result as the following form.

$$(b-d)z=b(z-f)\Rightarrow d=\frac{bf}{z}$$

Part 2 Disparity Estimation

Reference Paper

I implemented the task using ADCensus[1] cost. This work ranked #6 in the Middlebury v2 dataset, which mainly contributed in parallel computation through the whole process. I will briefly introduce the algorithm used in this work, and then compare the results I made.

Algorithm

Cost Computation

Absolute Difference Cost + Census Cost (ADCensus)

Absolute difference cost (C_{AD}) measures the color difference between the windows scanned in the left image and the right image. This directly obtains the color intensity information pixelwise.

[1] Ref: X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang. On Building an Accurate Stereo Matching System on Graphics Hardware. *GPUCV 2011*

The other measure is the census cost (C_{census}). Census cost encodes local information with relations between colors of the pixel into binary strings. It distinguishes different patterns with close absolute difference cost in RGB space. Therefore, this work considers both costs in order to discover the disparity for each pixel. After computation of both costs, normalize both costs to the range of [0, 1] by the formula

$$\rho(C, \lambda) = 1 - e^{-\frac{c}{\lambda}}$$

and the ADCensus cost is computed as

$$C(p, d) = \rho(C_{census}(p, d), \lambda_{census}) + \rho(C_{AD}(p, d), \lambda_{AD})$$

Cost Aggregation

Cross-based Cost Aggregation

This method is from another paper[2] proposed in 2009. The basic aggregation thought is identical but a slightly difference in aggregating directions and some local constraints.

In the previous paper, it only aggregates costs horizontally. In this 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. For example, iteration 1 and 3 aggregates horizontally and iteration 2 and 4 aggregates vertically. This is the implementation that differs from the previous work.

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, P_2

[2] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images. *IEEE TCSVT 2009*

depend on the color difference between query pixel and its predecessor both in left image and right image. The details are in the paper. After that, we 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 its 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. Otherwise, it will be labelled as a mismatch point.

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:

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

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 (i.e. color difference is minimized). 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 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.

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

Sub-pixel Enhancement

Sub-pixel enhancement can efficiently reduce the errors cause by discrete disparities. This step is found in the previous work. 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

3 x 3 median filter smoothes the interpolating results and get our final results.

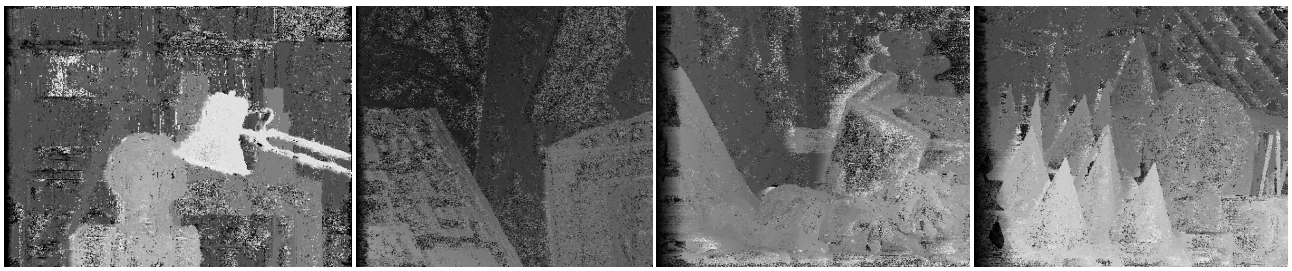
Bilateral Filter

The comments in this paper says that we may improve our results by adding bilateral filter when preprocessing the input image. Due to the cost is greatly dependent on the color intensity, some subtle color changes can cause a great difference. Thus, bilateral can help us smooth those small regions but remain the important edges. I implemented the bilateral filter by TA's code in HW1, and check if it do improve our results.

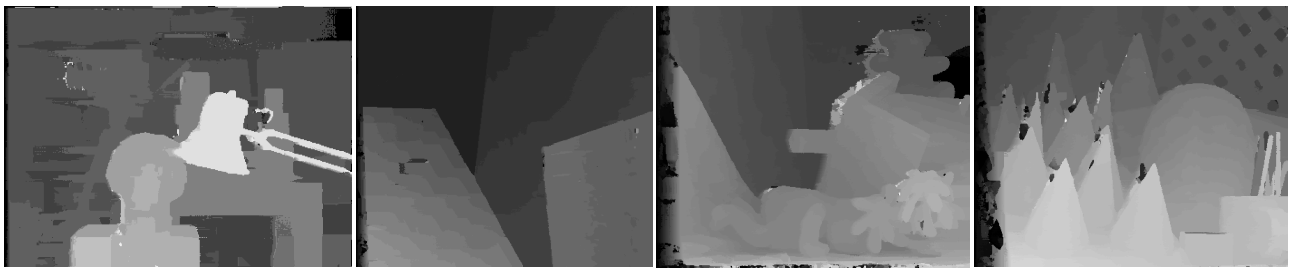
Results

We discuss about bilateral filter later. The results below does not include implementation of bilateral filter.

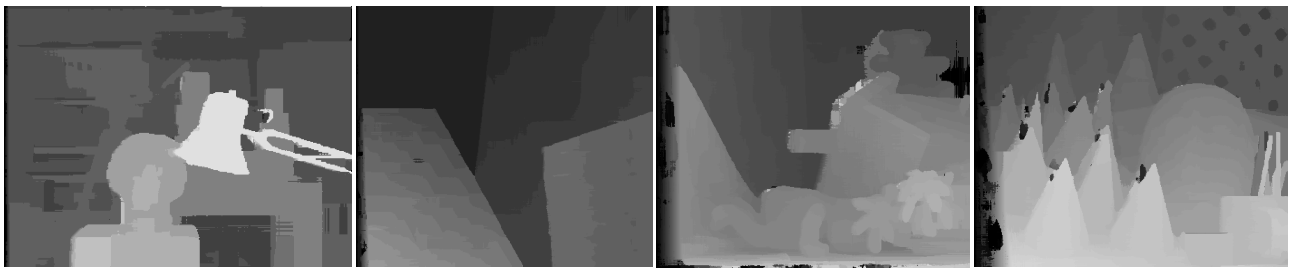
ADCensus cost



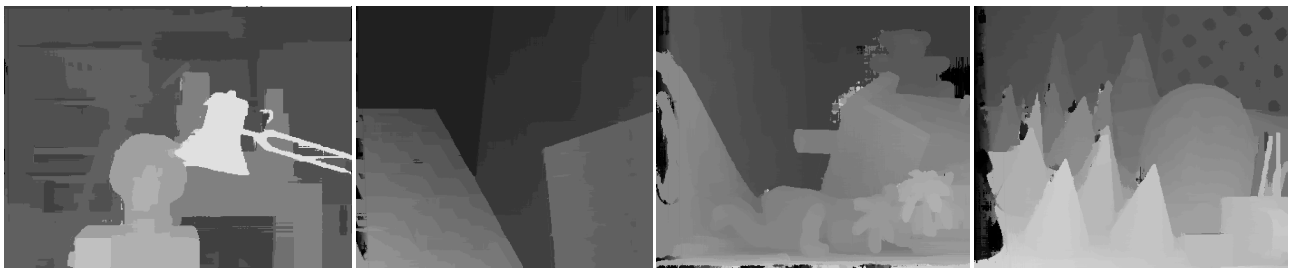
Cross-based Aggregation



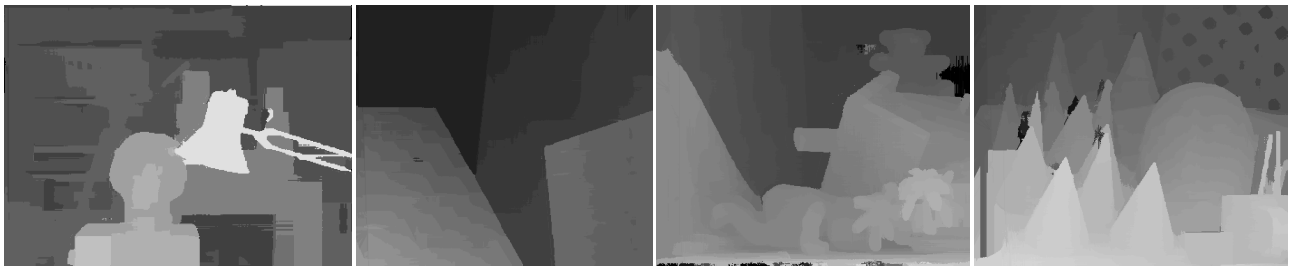
Scanline Optimization



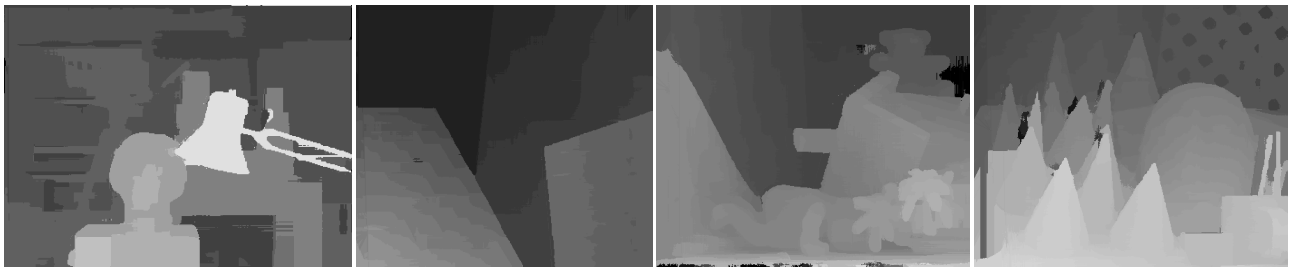
Iterative Region Voting



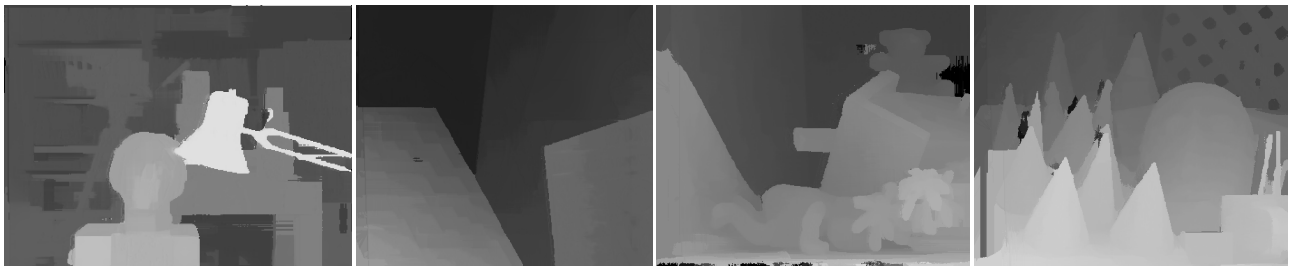
Proper Interpolation



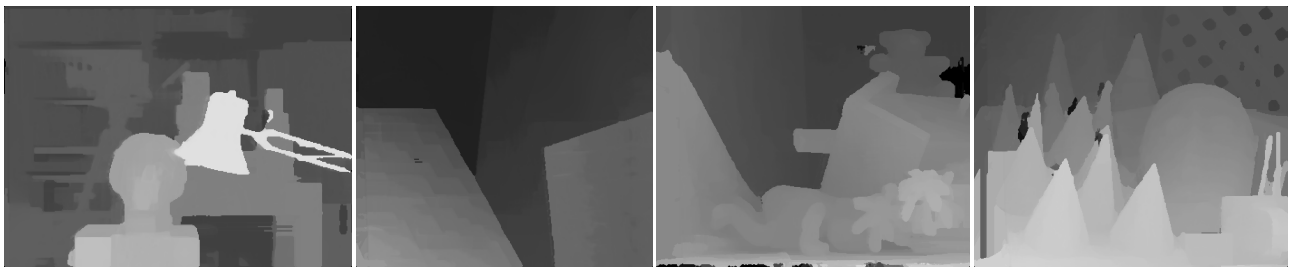
Depth Discontinuity Adjustment



Sub-pixel Enhancement

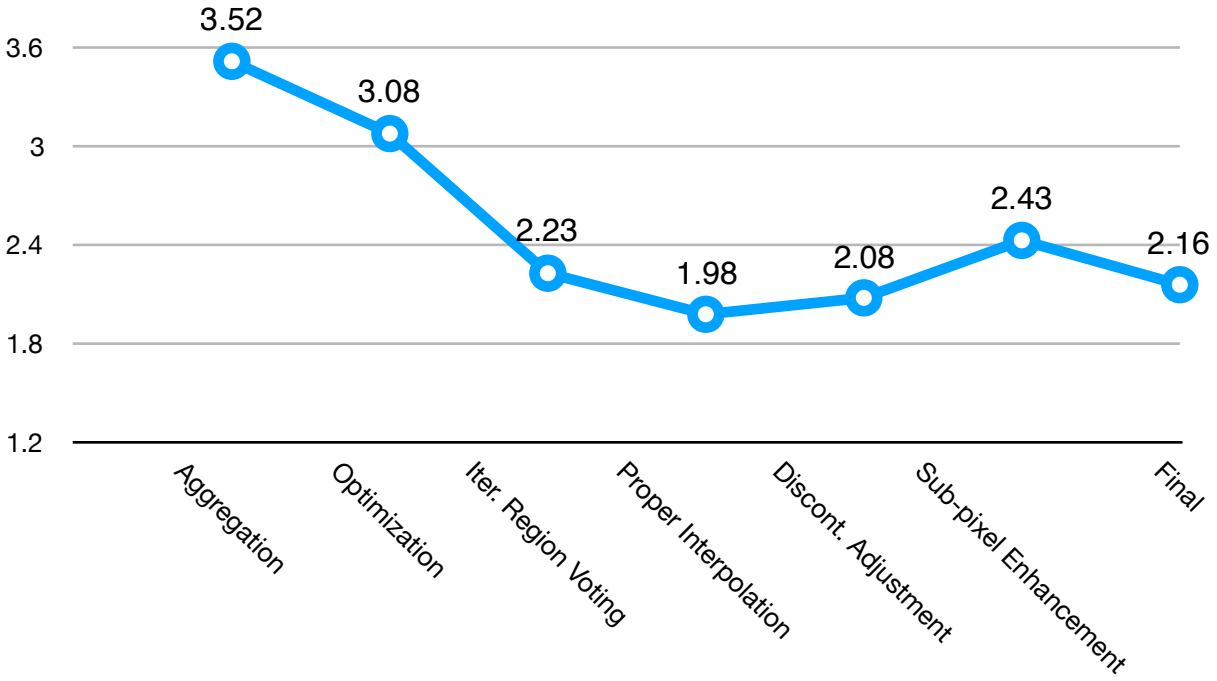


Final Results

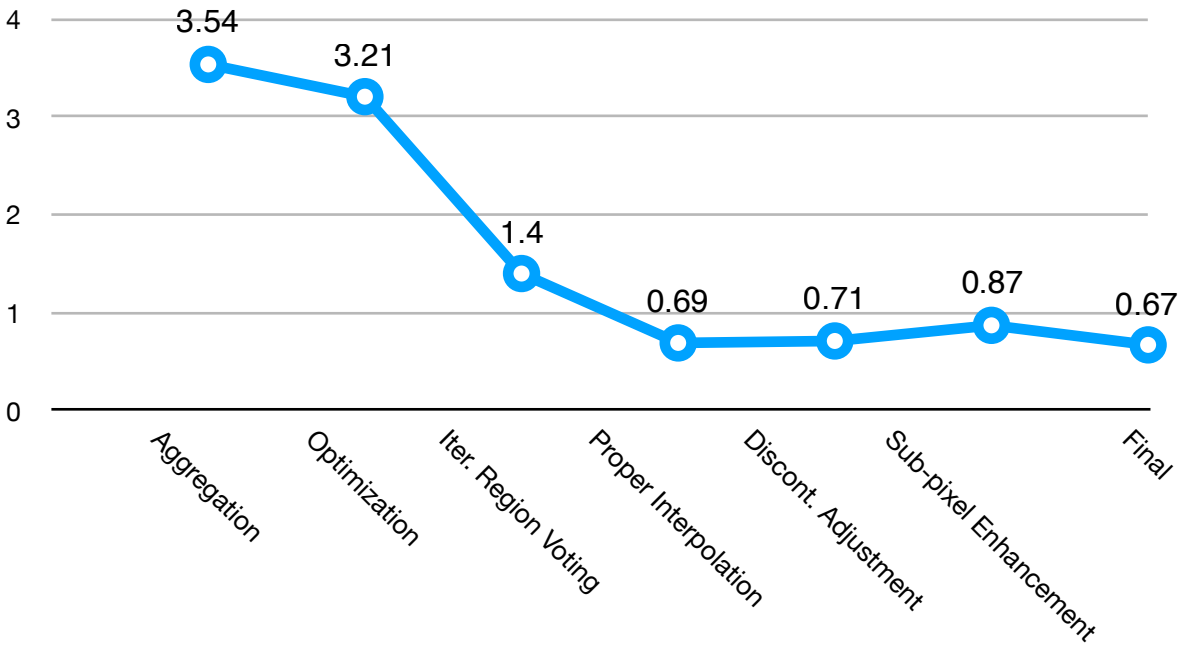


Evaluation Metric (Bad Pixel Ratio) During the Process

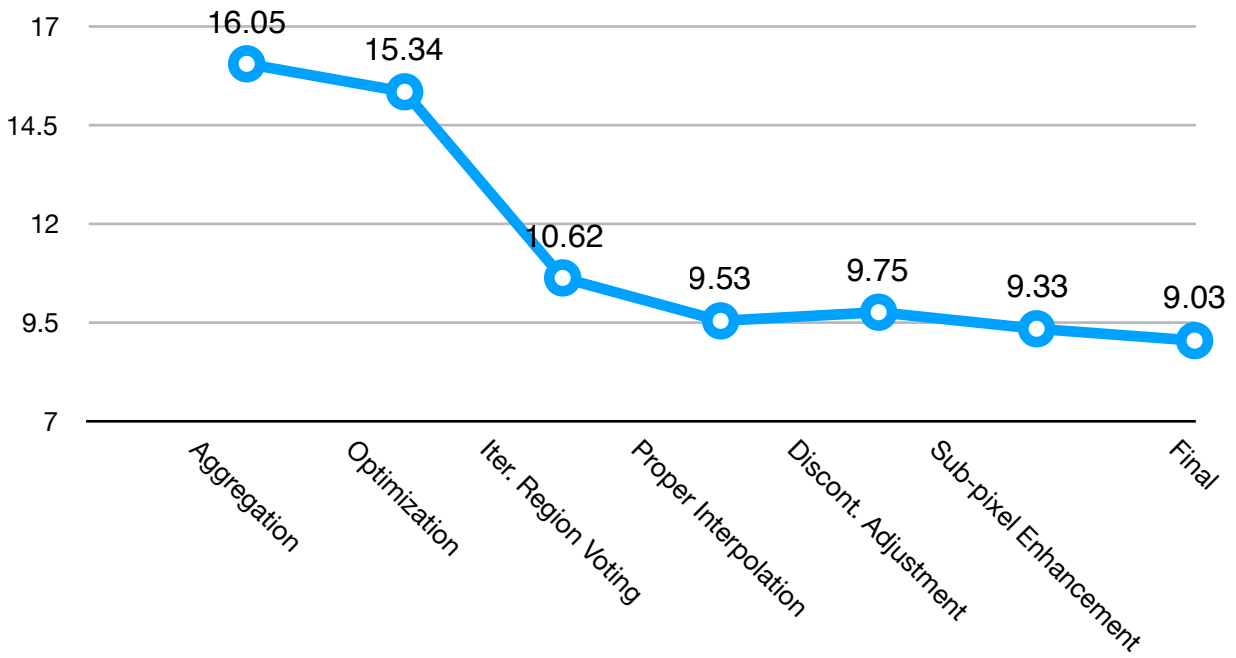
Tsukuba



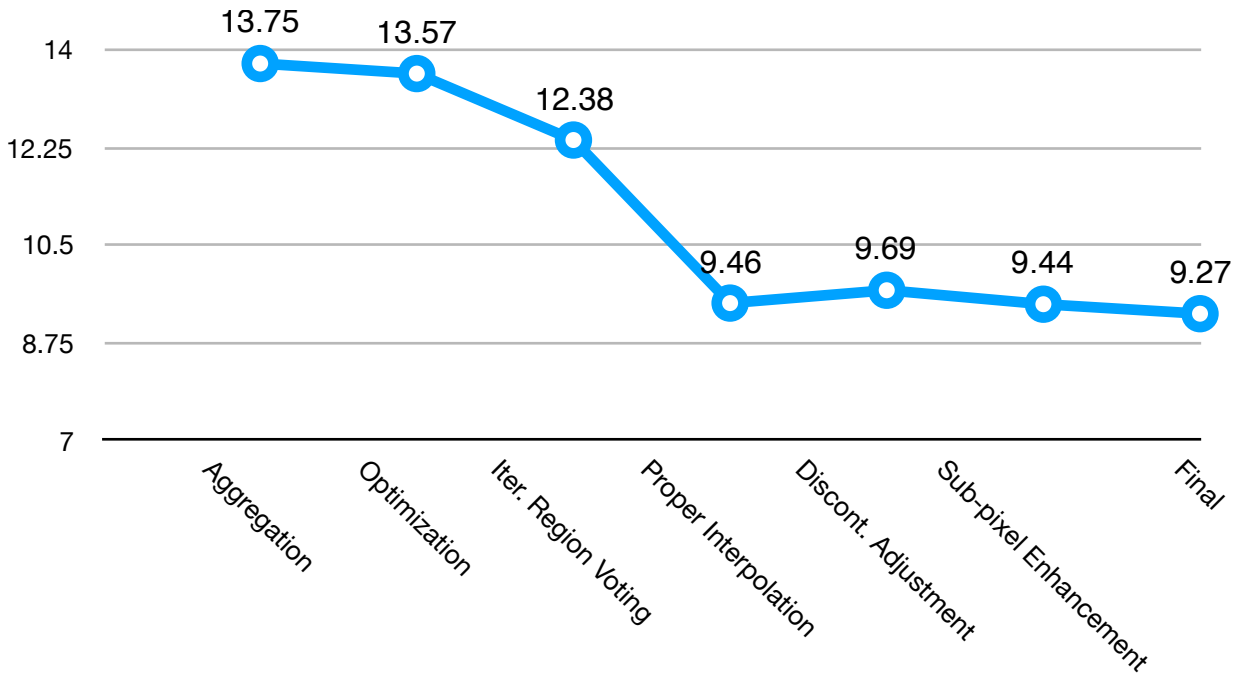
Venus



Teddy



Cones



Bad Pixel Ratio (Final)

Tsukuba	Venus	Teddy	Cones	Average
2.16%	0.67%	9.03%	9.27%	5.28%

Runtime

Tsukuba	Venus	Teddy	Cones
1314 sec	2345 sec	4563 sec	3229 sec

Discussion

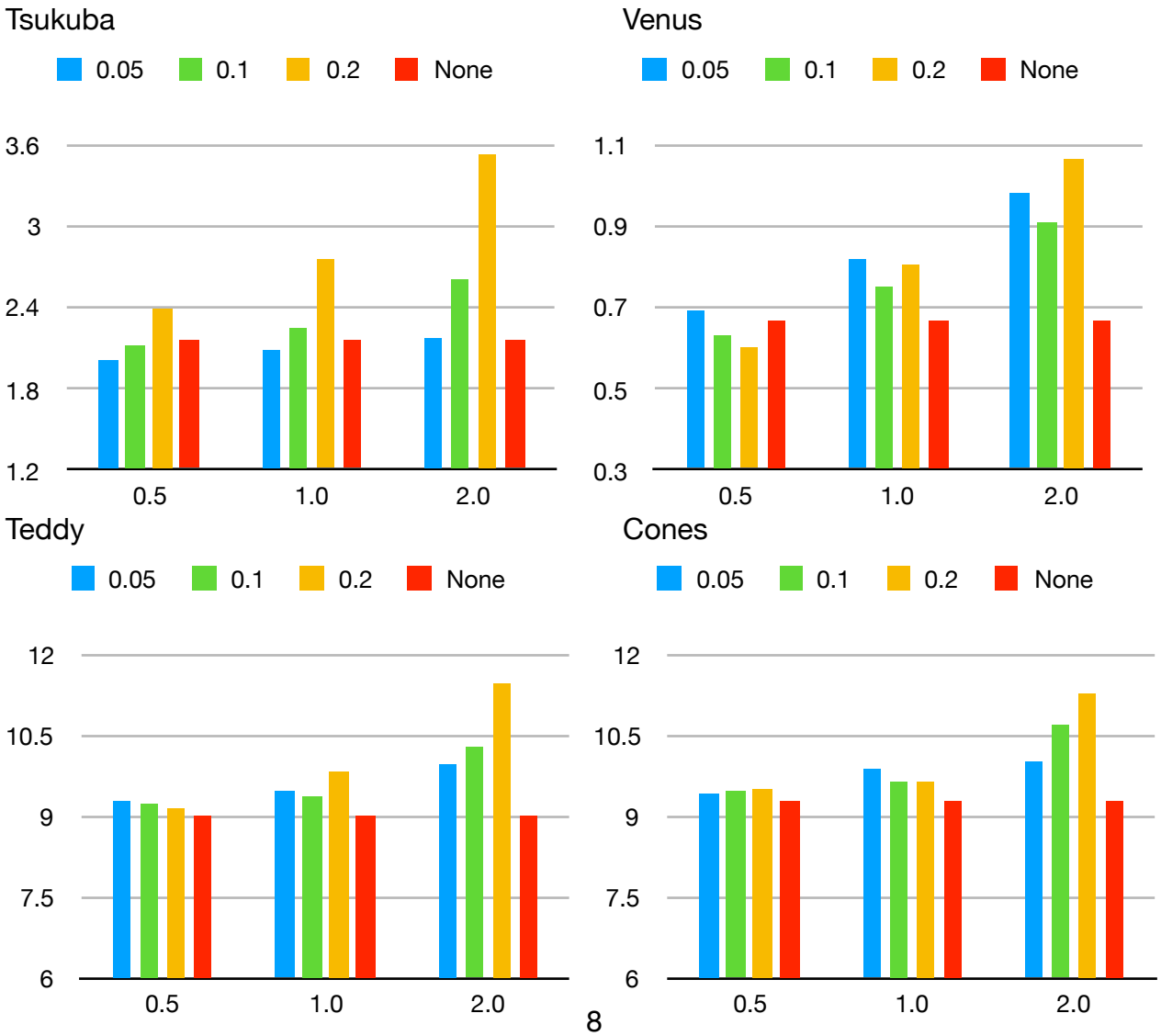
Although I implemented all the methods in the paper, it seems to have some differences between my results and theirs. Maybe some computational steps such as aggregation part has some minor mistakes that I did not notice. Therefore, I think that I could fix it and reduce the bad pixel ratio further more.

The runtime is not quite good enough. I will managed to speed up the performance by lowering the computational complexity.

Notice that the bad pixel ratio curve during the process, there exists a peak while performing depth discontinuity adjustment. It seems that it generates discrete interpolation beside edges and probably makes the edges more inaccurate in some cases. Somehow the sub-pixel enhancement does not improve the accuracy while it smoothes the discontinuity areas. One assumption is that it greatly influence other areas just as the figure shown above. It do reduce the discontinuity but it also affect other areas, leading to a slightly worse results.

Another discussion is that whether bilateral filter is available for preprocessing. The following shows the results and the hyperparameter I chose for spatial and range kernel, respectively.

Somehow similar to HW1, I chose sigma_s = **{0.5, 1, 2}**, sigma_r = **{0.2, 0.1, 0.05} * 255**.



"None" represents the results without bilateral filter, compared to others with preprocess of bilateral filter. From the result above, we can observe that there may have some improvement by adding bilateral filter (using the image itself as guided image). Some results in tsukuba may have better results than before. Others, however, do not show the same outcomes as tsukuba, especially sigma_s = 2.0 or sigma_r = 0.2. Both of them generate worse results than previous ones obviously. Hence, I had my final results without the bilateral filter.