Lab Assignment - Stereo Matching

Kaiyue Shen

November 15, 2020

1 Winner-takes-all

We first use the given code to rectify a pair of images. Since images are rectified, disparity only happens at the horizontal line. For each pixel, we want to find the disparity that minimizes the SSD/SAD between pixel pairs. To make the computation faster, we use the average filter to replace the "for" loop while computing the SSD/SAD in a certain window. We try different average filter window sizes: $3 \times 3, 7 \times 7, 15 \times 15$, the disparity map results are shown in figure 1,2 and 3. We can see that with increasing window size, small noise are gradually removed and the disparity map becomes smoother.

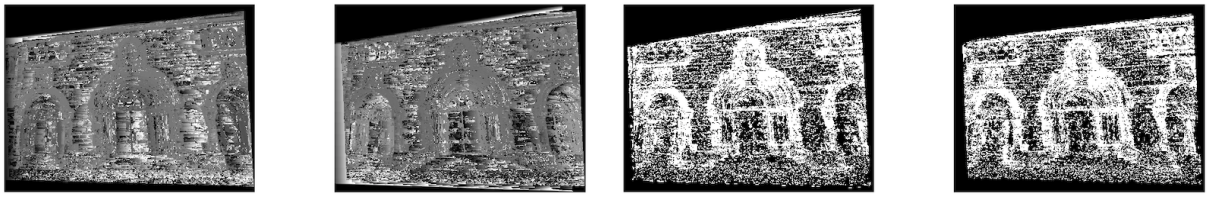

Figure 1: Winner-takes-all, window size: 3×3 (a) Disparity map (b) Mask

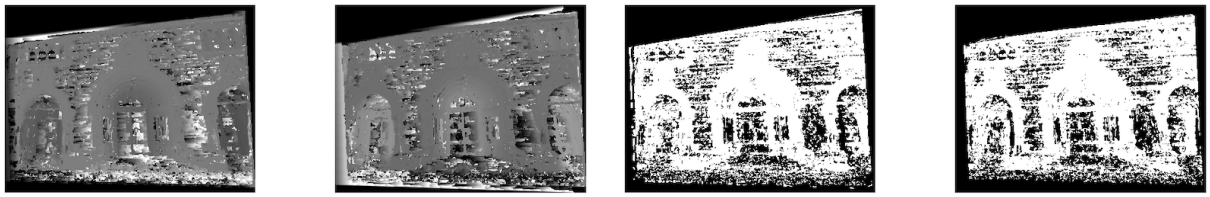

Figure 2: Winner-takes-all, window size: 7×7 (a) Disparity map (b) Mask

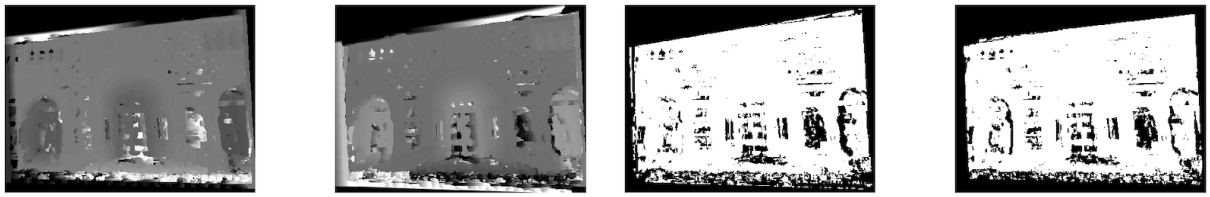

Figure 3: Winner-takes-all, window size: 15×15 (a) Disparity map (b) Mask

2 Graph-cut

The disparity computation problem can also be formulated as a graph labeling problem. Each pixel corresponds to a graph node and each disparity to a label. We want to find a labeling that minimizes an energy function, so we need to define this function. The first cost term is SSD value, corresponding to the first criterion: matching pixels should have similar intensities. The second term is the smoothness cost, corresponding to the second criterion: most nearby pixels should have similar disparities. The graph-cut result is shown in figure 4. Comparing figure 1 and 4, we find graph-cut performs better than winner-takes-all: with window size 3×3, graph-cut result is already smooth enough.

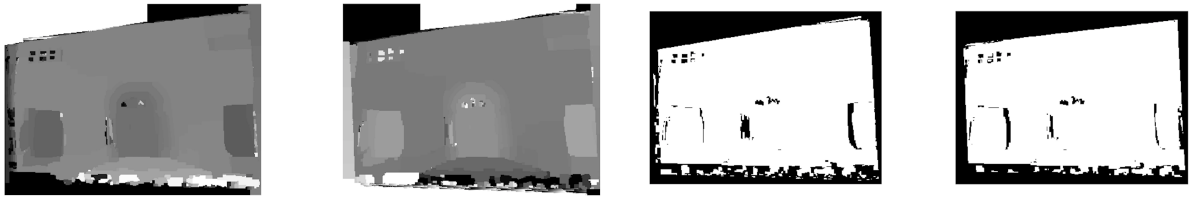

Figure 4: Graph-cut, window size: 3×3 (a) Disparity map (b) Mask

3 Generating a textured 3D model

From figure 5, we can further verify that graph-cut performs better than winner-takes-all. Although winner-takes-all uses larger window size than that used in graph-cut, the final reconstruction of graph-cut looks more complete with little noise.

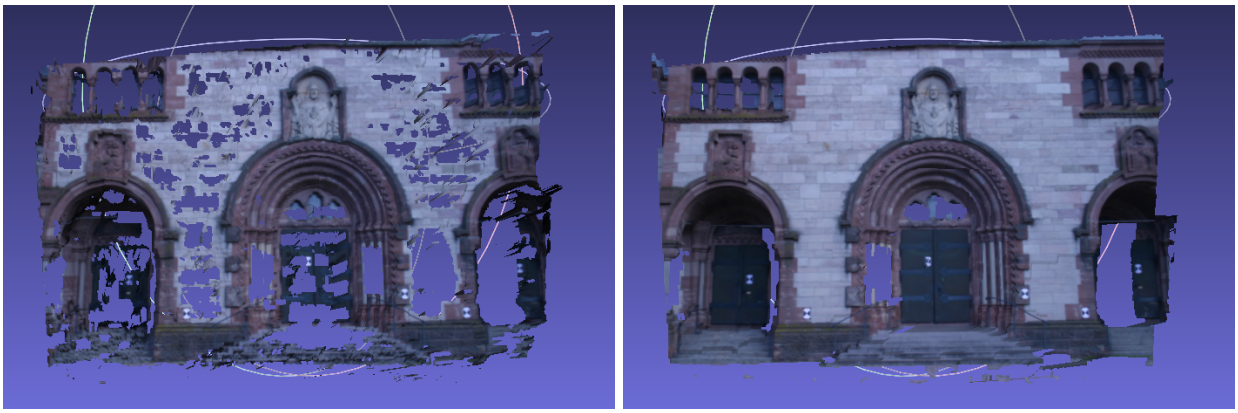

Figure 5: 3D model: (a) Winner-takes-all, window size 15×15 (b) Graph-cut, window size 3×3

4 Automate disparity range

We propose two automatic disparity range computing methods. One is to first manually select several pair of points and then compute the disparity between these points, use the $\pm \max \{abs(99\%\text{quantile}), abs(1\%\text{quantile})\}$, as the final lower-bound/upper-bound. Another one is fully automatic, we first extract the SIFT features and perform matching. Then we use RANSAC to further improve the matching result and compute the disparity for inliers. Finally we use the same formula above to compute the disparity range. Using given image pairs, the range using these two methods are similar $[-11, 11], [-12, 12]$, both smaller than the fixed one $[-40, 40]$, which means we can save much computation time. Comparing quality results in figure 4 with 6 and 7, we can see that although the disparity search range are shortened, the final result look similar.

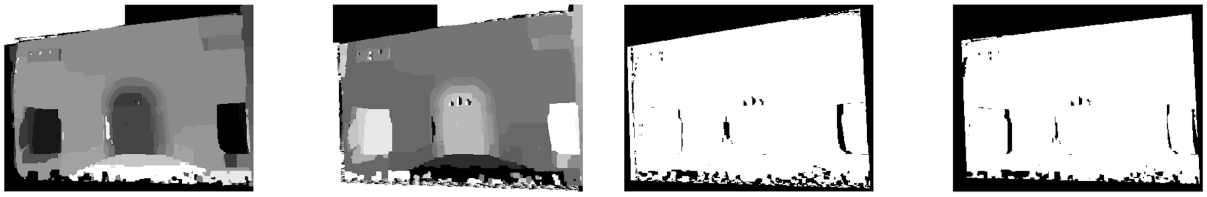

Figure 6: Graph-cut, window size: 3×3, Method1 (a) Disparity map (b) Mask

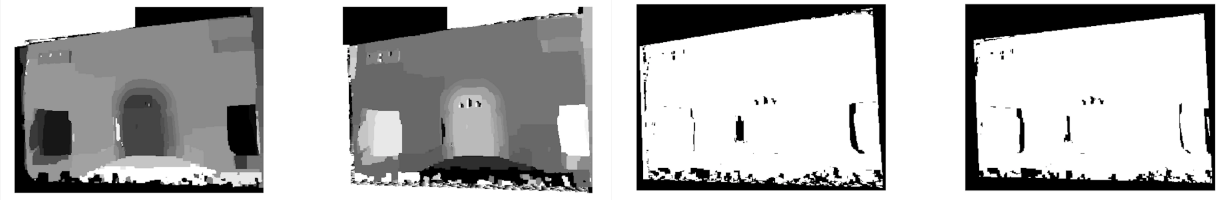

Figure 7: Graph-cut, window size: 3×3, Method2 (a) Disparity map (b) Mask