Computer Vision Exercise 6: Stereo Matching

Soomin Lee (leesoo@student.ethz.ch)

November 13, 2020

1. Disparity Computation: winner-takes-all

Given two images from different viewpoints, we first need to rectify the images so that the epipolar lines become parallel. For rectification, the code is already given. In a nutshell, we calculate fundamental matrices using SIFT feature matching, and transform the images to the standard stereo setup. After the rectification process, we only have to go through the horizontal line to estimate the optimal disparity d^* that minimizes the SSD or SAD between image patches. In this report SAD was used:

$$d^* = arg\min_x \Sigma_{i \in patch} |I_0(p_0 + i) - I_1(p_0 - [x; 0] + i)|. \tag{1}$$

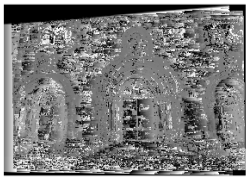
In order to speed up the procedure, the calculation for a fixed range of disparity was implemented as follows: for each d in the range (a) shift the entire image horizontally by d (b) calculate the absolute difference between the image pair (or square of the difference if one wants to use SSD) (c) convolve it with an averaging box filter (d) save d that yields the smallest value in the previous step for each individual pixel.

The results are plotted in Figure 1 for three different sizes of the averaging box filter. The figures on the left illustrate the disparity maps, while the figures on the right show the boolean masks of the pixels with disparity values that agree between the image pairs i.e. if the disparity values obtained for each image differ less than a certain threshold it is set as 1, otherwise 0. For these particular figures, the threshold was set to 8. As the size of the filter increases, one can notice that the disparity maps become smoother since a filter of a bigger size averages the values over a broader region. Also, the masks grow as the size of the filter increases, which can be understood intuitively that as the differences between the pixels become less extreme it's more likely that pixels have similar disparity values and thus more pixels tend to agree between the image pairs.

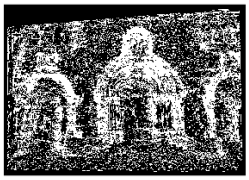
2. Disparity Computation: graph-cut

We can also compute the disparity using the graph-cut method. By considering each pixel as a graph node and each disparity to a label of a node, the algorithm tries to find a label for each node that minimizes the cost which is a combination of a SSD value and a penalty term for neighboring pixels having different labels. SSD values can be calculated in a similar manner as in winner-takes-all method: shift one image and calculate square of the difference between corresponding pixels, and replace the summation within the local patch with the convolution of the image with an averaging filter. The implementation of the second term that enforces smoothness between the neighboring pixels is given in the sample code, and eventually it is implemented using `GCMex` package.

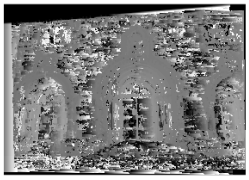
The results are shown in Figure 2, following the same format as Figure 1. The tendency of smoothness is also the same as before i.e. bigger the filter size, smoother the disparity. The more interesting thing to observe is the difference between Figure 1 and Figure 2. The disparity maps obtained using the graph-cut method are significantly smoother than those obtained using the winner-takes-all method, and masks cover a larger portion of the images accordingly. This is indeed a reasonable result since the graph-cut method includes smoothness term in the cost function. One can also say that the method relies on the assumption that nearby points have similar disparity values in the first place.



(a) Filter size 3×3, disparity map

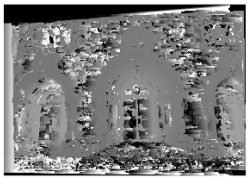


(b) Filter size 3×3, mask



(c) Filter size 5×5, disparity map

(d) Filter size 5×5, mask



(e) Filter size 7×7, disparity map

(f) Filter size 7×7, mask

Figure 1: Disparity maps (left) and masks of the disparity maps that those two maps agrees on up to threshold (right) for different filter sizes, obtained using the *winner-takes-all* method.

(a) Filter size 3×3, disparity map

(b) Filter size 3×3, mask

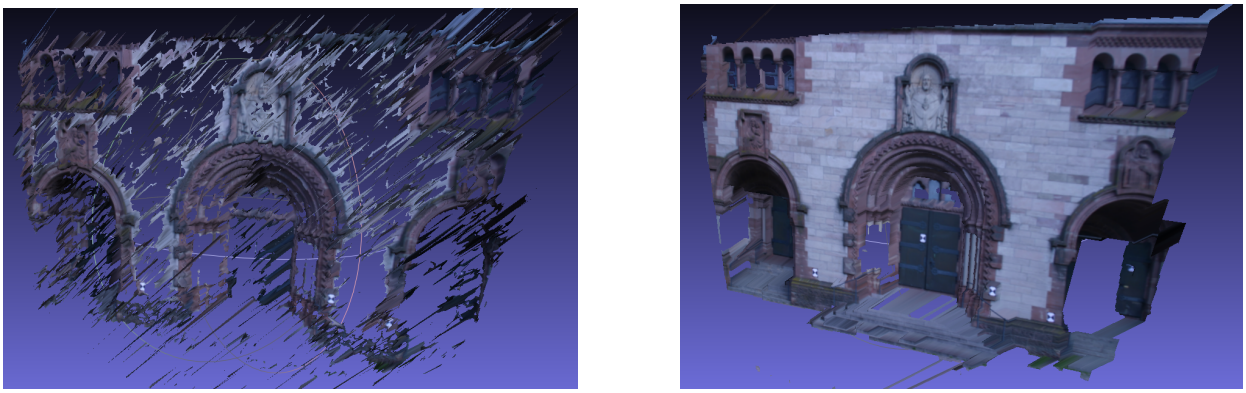
(c) Filter size 5×5, disparity map

(d) Filter size 5×5, mask

(e) Filter size 7×7, disparity map

(f) Filter size 7×7, mask

Figure 2: Disparity maps (left) and masks of the disparity maps that those two maps agrees on up to threshold (right) for different filter sizes, obtained using the *graph-cut* method.



(a) Winner-takes-all (b) Graph-cut

Figure 3: Textured 3D models for each method.

3. Textured 3D Models

Figure shows the textured 3D models, which are visualized using `Meshlab`. As one can expect from the smoothness of the disparity maps from the previous figures, the model generated with the graph-cut method looks significantly smoother also in 3D. In the model generated with the winner-take-all method, the disparity value of each pixel changes too sharply even for the neighboring pixels that the 3D model is harder to recognize.

4. Automated Disparity Range

Until now, the disparity range that the algorithm goes through was fixed arbitrarily as -40 to 40. However, we can also adjust the range to improve the quality of disparity estimation. Figure 4 shows the manually selected point correspondences for determining disparity range. Since the two images are already rectified, one can get the disparity by calculating the difference in x-coordinates between each pair. For the points in Figure 4, the maximum disparity was 11, and the minimum disparity was -11. Therefore, the disparity map was calculated once again using the graph-cut method but with the range -11 to 11 instead of -40 to 40. The filter size was fixed as 7×7 in this case, and results are shown in Figure 5 and Figure 6.

Figure 4: Manually selected point correspondences.

(a) disparity map (b) mask

Figure 5: Disparity maps (left) and masks of the disparity maps that those two maps agrees on up to threshold (right), obtained using the graph-cut method with an adjusted disparity range.

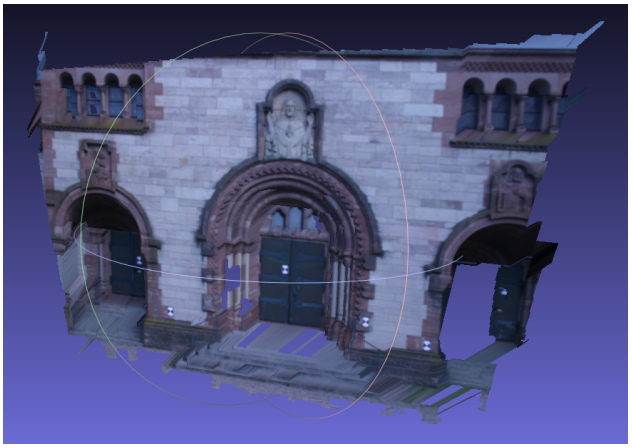


Figure 6: Textured 3D model for the graph-cut method with an adjusted disparity range.

The disparity map shows more changes in the color not because it is less smooth but because it has higher resolution in terms of the disparity change as we limited the total range. One can confirm it with the mask i.e. Figure 5b has even less black areas than Figure 2f. Also, in Figure 6, the 3D model doesn't have points that are extruding to the front or to the back anymore. Moreover, it has better reconstruction quality as some of the parts that were missing in the previous figures are now filled in reasonably.