Computer Vision Assignment 6

Luca Ebner (ebnerl@student.ethz.ch)

November 14, 2020

In real world applications it is often of great interest to estimate the distance between the camera and the objects in the image. To do so, one can use stereo cameras and geometric properties. The key idea here is to take a photo of an object with two cameras next to each other from the same angle. If we then compare the two images, we notice that objects closer to the camera setup are horizontally shifted by a larger amount of pixels than objects that are far away.

1. Disparity Computation

The horizontal shift of objects between two images of a stereo camera setup is denoted as the disparity. If we compare the pixels of the left image with the right and note their disparity, we can create another image consisting of the disparity values, the so-called "disparity map". The disparity in turn is directly related to the distance from the objects to the camera as seen in Figure 1.

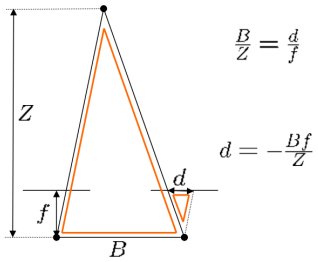

Figure 1: Geometric relation between the disparity and image depth. B is the baseline offset between the cameras, Z is the distance to the object, f is the focal length and d is the disparity

To find the disparity map, one first has to match the pixels between the two images. In this exercise, the pixels were matched by minimizing the sum of squared distances (SSD) of the greyscale images. Note that for the best match for a pixel in image 1 the search can be limited to its epipolar line in image 2, which in case of a perfect stereo setup is the same horizontal line (pixel height) as in image 1! On top of that, the best match is only searched for within a given range of disparity. (In case of this exercise, `d = -40:40`.) Additionally, since the setup, rectification and other noise is never perfect, an average filter was applied. The results from the Winner-Takes-All disparity algorithm can be seen in Figures 2, 3, 4 and 5.

Figure 2: Original RGB images from the left and right camera (rectified).

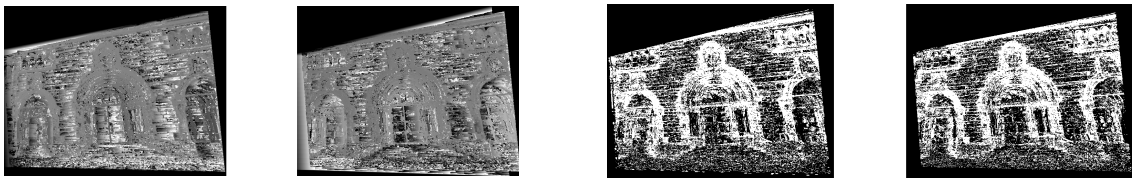

Figure 3: Disparity maps with average filter of size 3×3

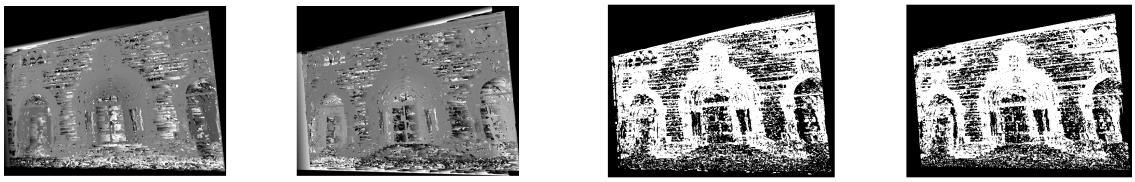

Figure 4: Disparity maps with average filter of size 5×5

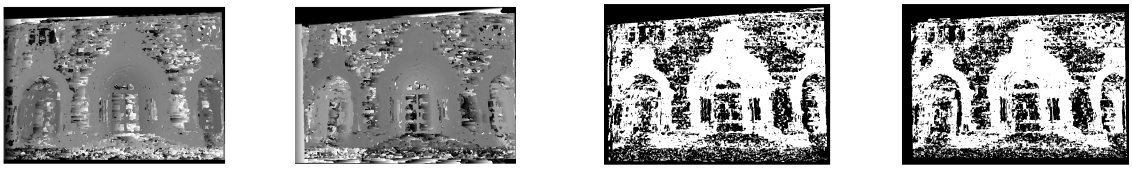

Figure 5: Disparity maps with average filter of size 7×7

We can easily see that the bigger the filter size, the more "washed out" the disparity maps are. On the other hand, when we choose a smaller filter size, the results are sharper but also more noisy. If we check the disparity matching masks on the right we can see that the choice of the filter size is a trade off between disparity map sharpness and consensus between the left and right disparity map. Smaller filter size leads to sharper results but less consensus.

2. Graph-Cut

Another way to compute disparity maps is by seeing the disparity maps as a graph labeling problem. For this, one defines a cost function E and tries to find a labeling $f : P \to L$ that minimizes the cost:

$$E(f) = E_{data}(f) + \lambda E_{smooth}(f) = \sum_{p \in P} D_p(f_p) + \lambda \sum_{p,q \in N} S(f_p, f_q), \tag{1}$$

Where P is the set of pixels, L is the discrete set of labels, $D_p(f_p)$ is the cost of assigning label f_p to pixel p, N is the set of neighboring pixels and $S(f_p, f_q)$ is the cost of assigning labels f_p and f_q to pixels f and q. In this exercise, the smoothness term was already given. The other term, $E_{data}(f)$ can be similarly computed with the SSD value as in the winner takes it all method.

The idea behind this cost function is to penalize assigning different labels to neighboring pixels, whereas this was not the case in the Winner-Takes-All method. The introduction of such a smoothness term leads to much more realistic disparity maps as seen in Figure 6.

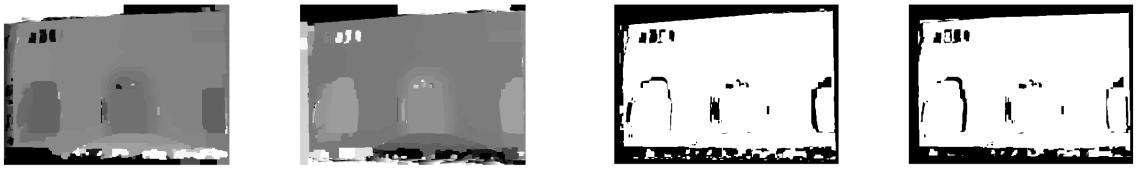

Figure 6: Disparity maps and consensus masks from the Graph-Cut algorithm.

It is clearly visible that this is a great improvement compared to the Winner-Takes-All algorithm. the disparity maps themselves appear very smooth and less noisy. Additionally, the disparity map consensus masks look less sparse as with Winner-Takes-All.

Generating a textured 3D model

Together with the obtained disparity maps and the camera information, a textured 3D model of the stereo image can be generated. The respective code framework to do so is already provided in the exercise. A comparison of the 3D models from the Winner-Takes-All and Graph-Cut algorithm visualized in MeshLab can be seen in Figure 7 and 8. Once again it becomes clear that the Graph-Cut approach delivers much more realistic results.

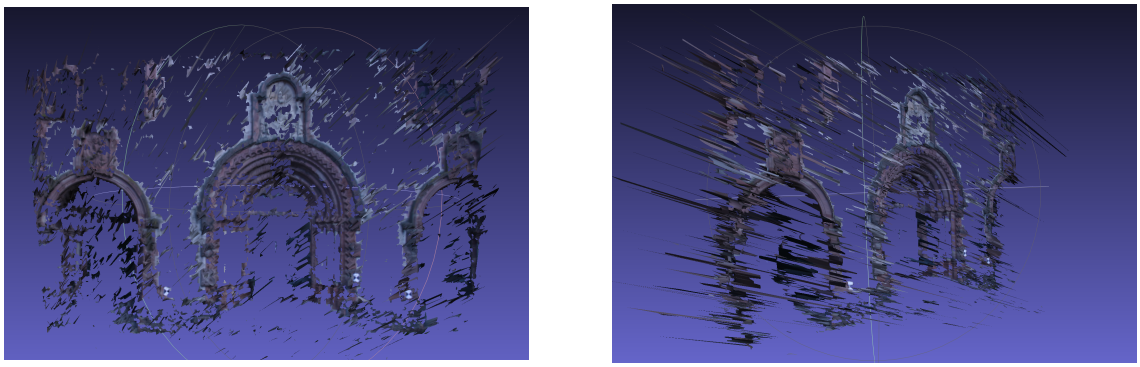

Figure 7: 3D model of Winner-Takes-All algorithm. Note that the there is a lot of noise in the depth mapping as already seen in Figure 3-5.

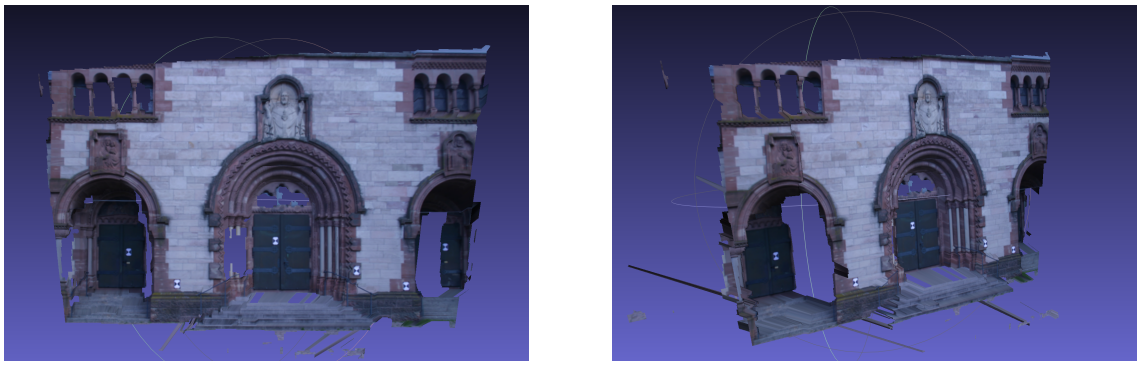

Figure 8: 3D model of Graph-Cut algorithm. Note that the noise in the depth mapping is now mostly gone because of the smoothing term in the cost function.

3. Automate Disparity Range

To reduce computation cost, it makes sense to estimate the range for valid disparity values instead of using a hardcoded value. For a meaningful estimation, one can take a few correspondence points and check for the biggest disparity. To cope with errors during the manual clicking of points, the computed maximum value of disparity is scaled with a chosen threshold, e.g. `scaleThresh = 2`. This way one can be sure that the correspondence points with the biggest disparity still lie within the scaled disparity range. In case of the images provided in the exercise, the estimated maximum disparity was ∼12 pixels. Together with the scaling threshold, the resulting disparity range was then found to be `d = -24:24`. That's almost only half the range compared to the hardcoded range of `d = -40:40`!

Note that, if done correctly, such an estimation can potentially reduce the amount of wrongly matched points. If we know the maximum disparity, we know that the true matches all lie within that range and other numerically good SSD matches out of that range are automatically excluded. In Figure 9 and 10 the resulting Graph-Cut disparity maps and 3D model can be seen.

Figure 9: Disparity maps and consensus masks of Graph-Cut algorithm together with disparity range estimation.

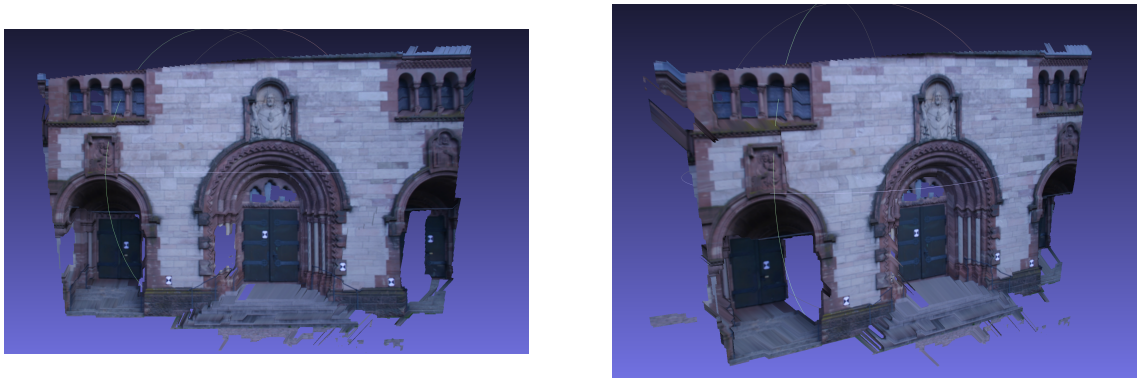

Figure 10: 3D model of Graph-Cut algorithm together with disparity range estimation.