# Computer Vision

Alessia Paccagnella

November 12, 2019

## 1 Stereo Matching

### 1.1 Disparity computation

To compute the disparity, I implement the winner-takes-all stereo using SSD. I use the code provided from the framework to rectify the two images.
Following the assignment instruction, for each of the disparities d, I shift the second image by d and calculate the SSD between the first image and the second shifted image. I use the function `fspecial` and `conv2` to convolve it with the box filter. After that I save the best disparity for every of the pixels.

Figure 1: Images 1 and 2

Figure 2: Images 1 and 2 rectified

The following images shows the disparity maps using windows of different sizes:

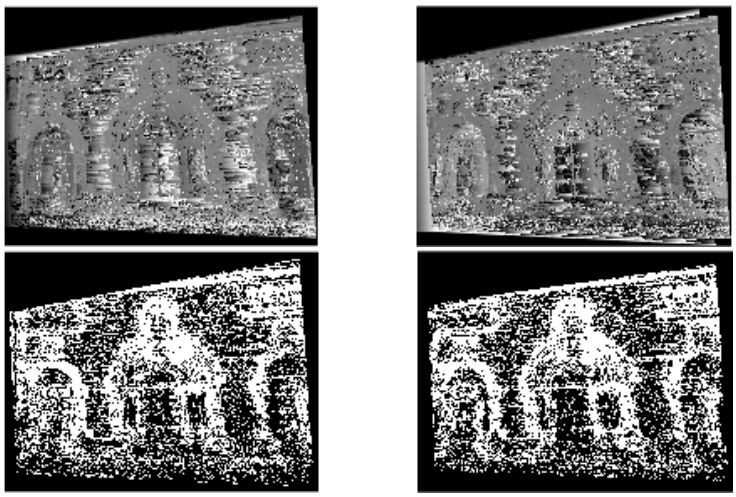

Figure 3: Disparity maps with window 3x3

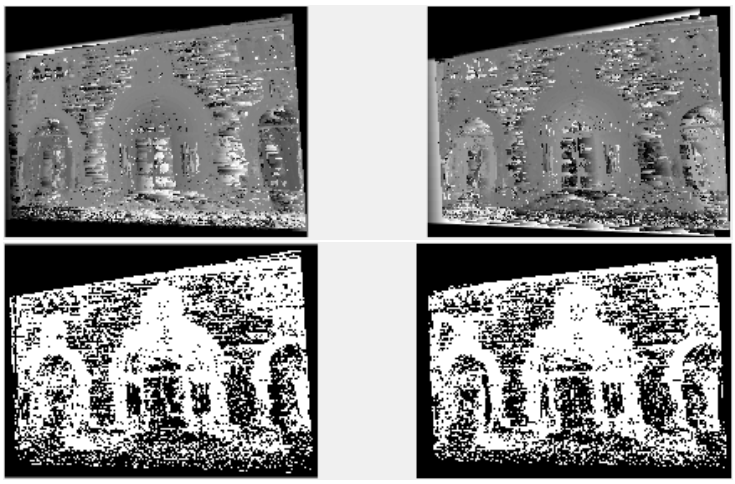

Figure 4: Disparity maps with window 5x5

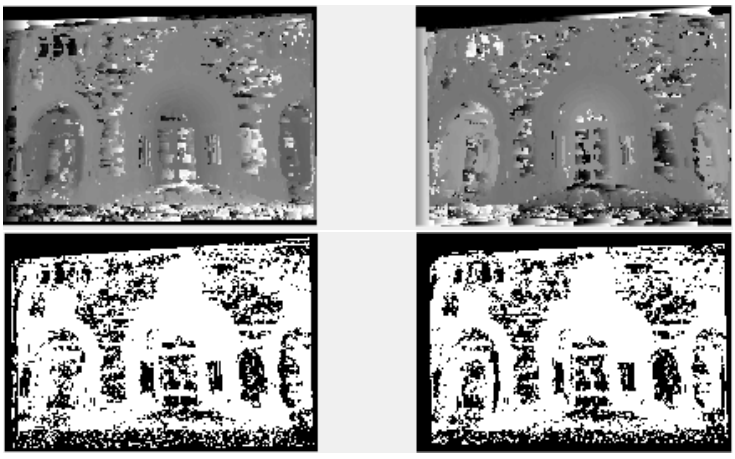

Figure 5: Disparity maps with window 10x10

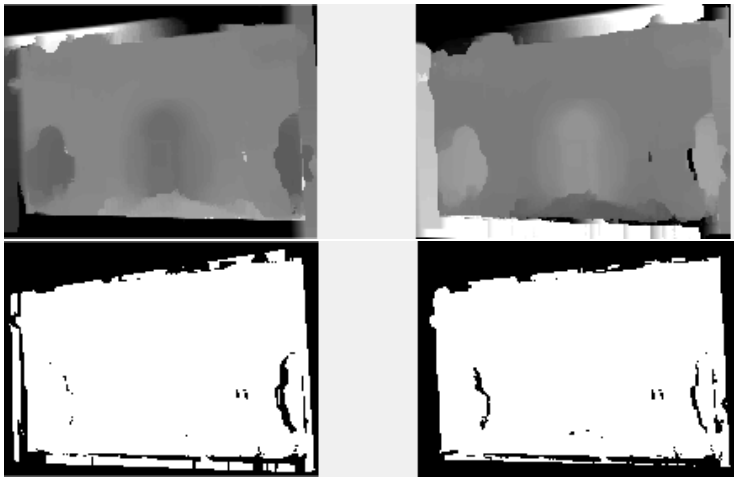

Figure 6: Disparity maps with window 50x50

I have noticed that the smaller the window taken is, the bigger is the number of discontinuities of depth. We can see it clearly from the 3D model reconstructed in subsection 1.3.

## 1.2 Graph-cut

The computation of disparities here is formulated as a graph labeling problem. Each pixel corresponds to a graph node and each disparity to a label. This algorithm wants to find a labeling that minimizes the energy function.
For this task I changed the data cost. I computed the SSD at each disparity for every pixel, convoluted it with the same box filter as before and stored the values obtained in a $m \times n \times r$ matrix. mxn is the size of the image and r is the number of disparities. The images shows the disparity maps using windows of different sizes:

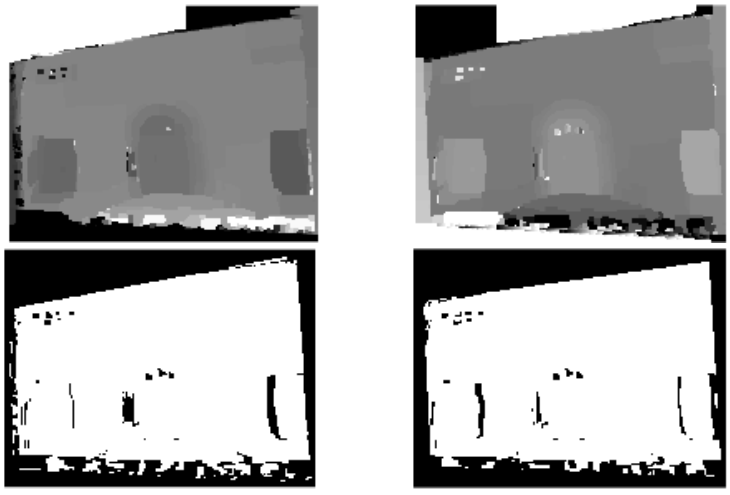

Figure 7: Disparity maps with window 3x3

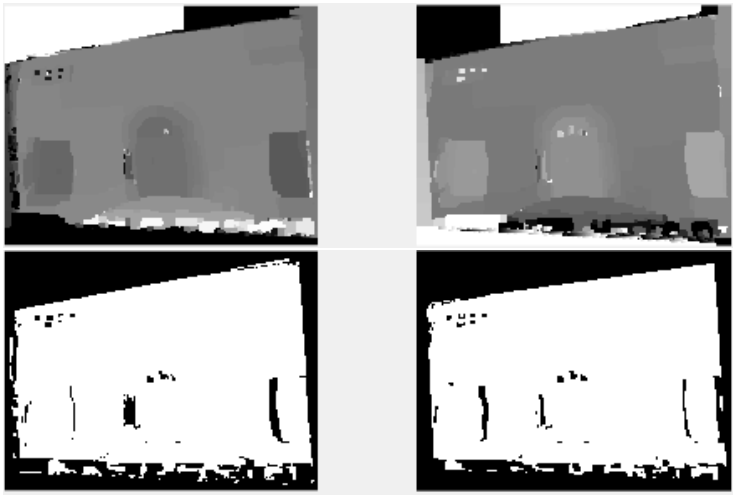

Figure 8: Disparity maps with window 5x5

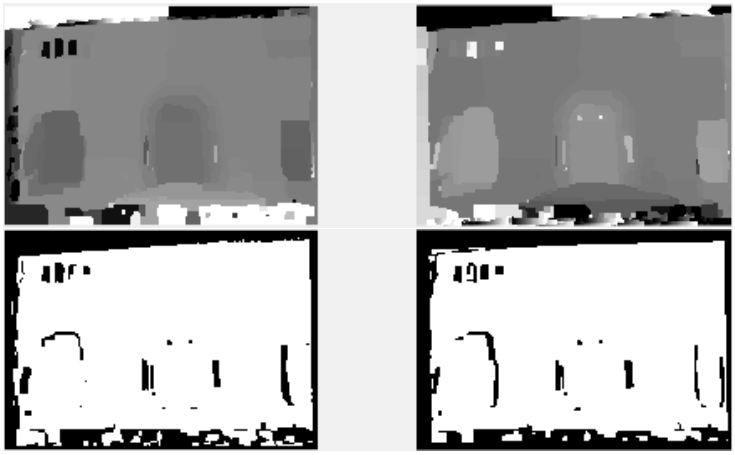

Figure 9: Disparity maps with window 10x10

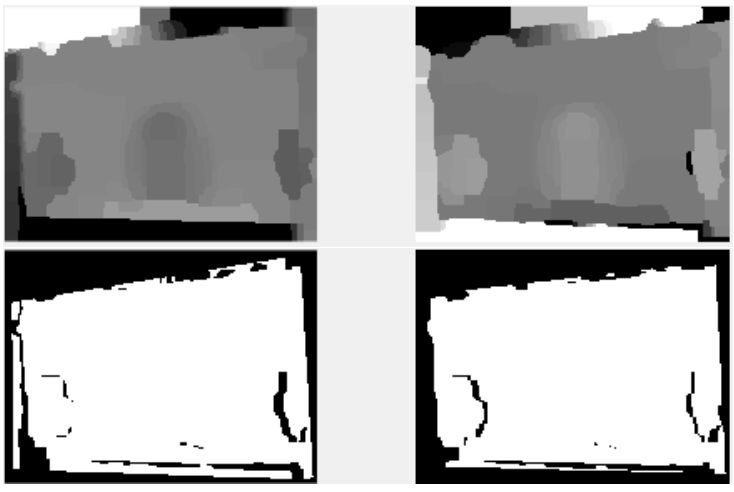

Figure 10: Disparity maps with window 50x50

## 1.3 Generating a textured 3D model

### 1.3.1 Stereo Disparity

For each pixel (x,y), the corresponding 3D point is computed, using the code provided from the function `generatePointCloudFromDisps`. The following images shows the results using windows of different sizes for both of the algorithms.

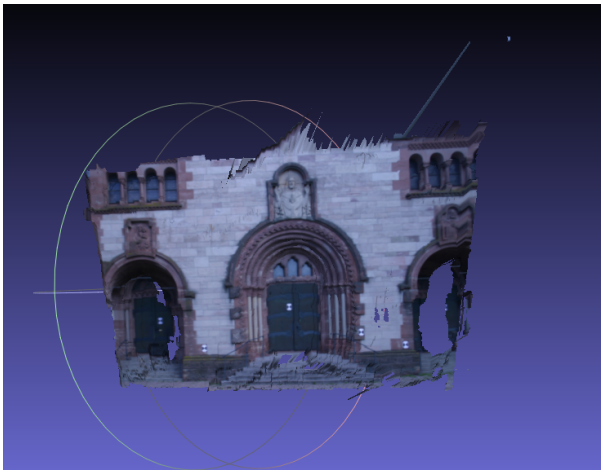

Figure 11: 3D model with stereo disparity, window 50x50

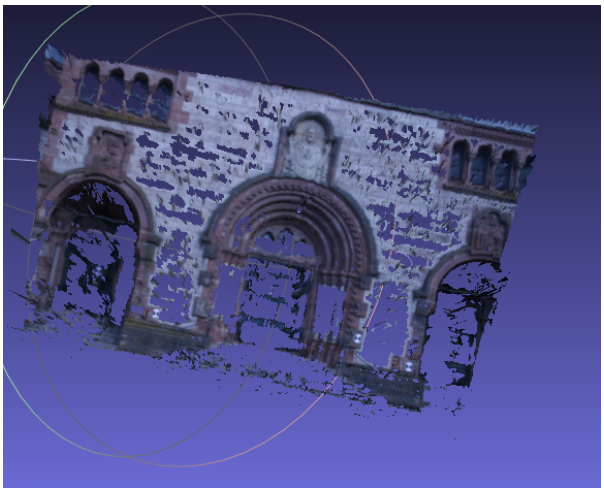

Figure 12: 3D model with stereo disparity, window 10x10

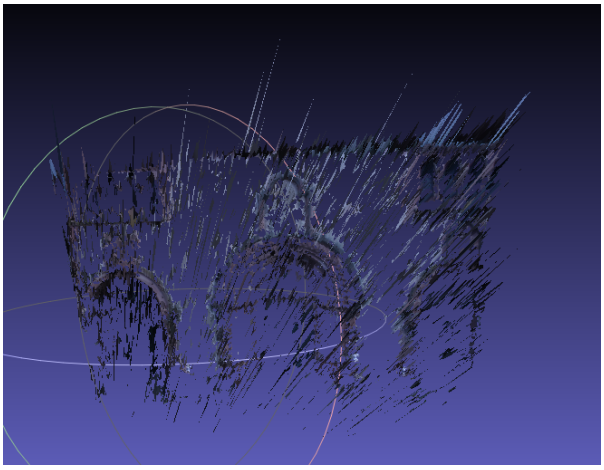

Figure 13: 3D model with stereo disparity, window 3x3

### 1.3.2 GraphCut

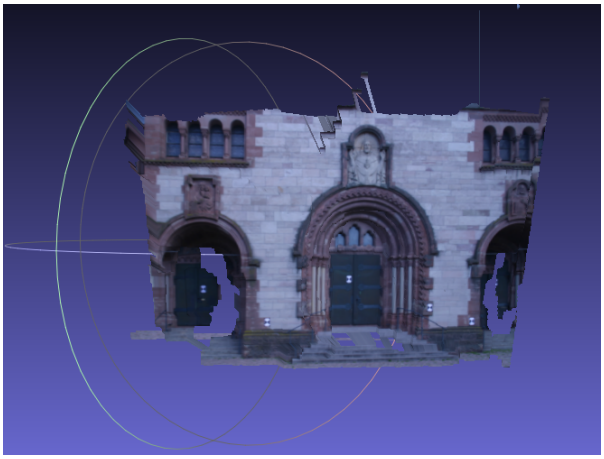

Figure 14: 3D model with GraphCut, window 50x50

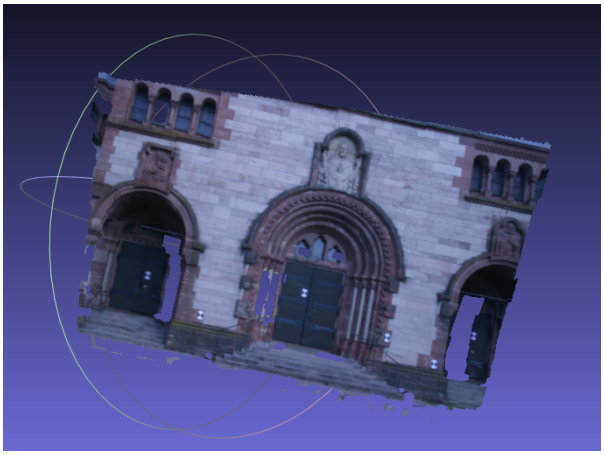

Figure 15: 3D model with GraphCut, window 10x10

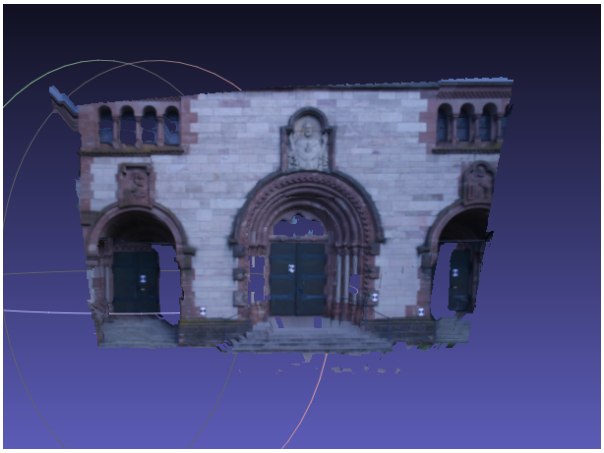

Figure 16: 3D model with GraphCut, window 3x3

The results obtained with CG are way better than the ones obtained with the winner-takes-all method, even when we use a small window (3x3 case). In that case, for the winner-takes-all we get a very bad result.
We can see that for the CG there are very small discontinuities, mainly at the doors, cause they change in depth. In those areas that vary a lot in term of depth, the algorithm does not know which disparity to give. When using a bigger window though, we still obtain a good result even with the winner-takes-all approach.

## 1.4 Automate disparity range

I wrote the lines of code that ask to click the corresponding point and then calculate the disparity range based on those points. I computed the disparity range to be from the minimum to the maximum disparity within -40 and 40. With this method, for the GraphCut I obtain the following results:

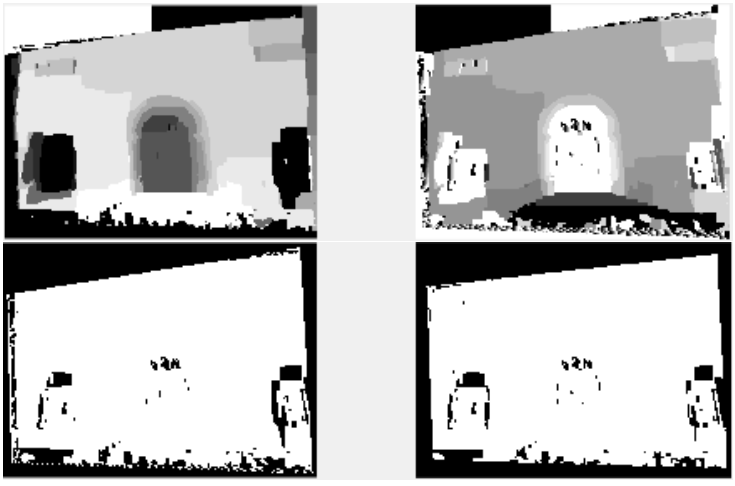

Figure 17: Disparity maps with GraphCut, window 3x3

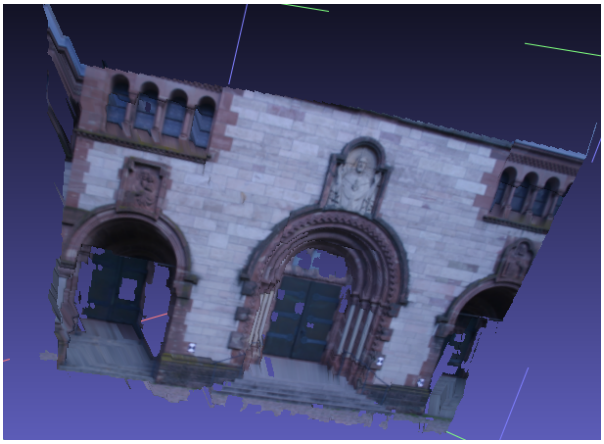

Figure 18: 3D model with GraphCut, window 3x3

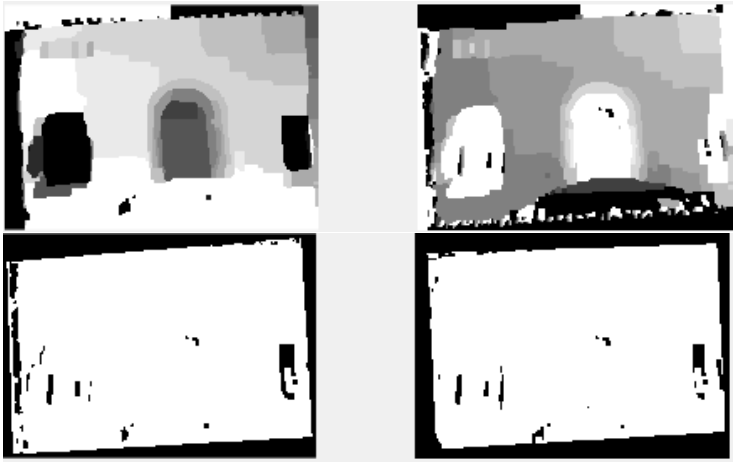

Figure 19: Disparity maps with GraphCut, window 10x10

Figure 20: 3D model with GraphCut, window 10x10

Figure 21: Disparity maps with GraphCut, window 50x50

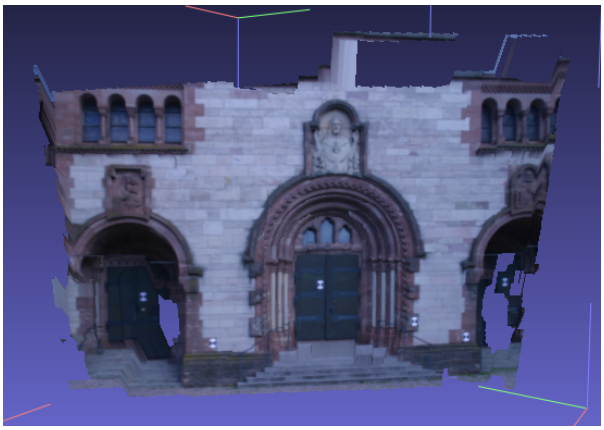

Figure 22: 3D model with GraphCut, window 50x50

It can be seen from these images that we obtain a better result for the doors and the stairs (generally, for parts with different depth) by using a bigger window. In this trials indeed, 50 is the best result obtained.