Computer Vision Assignment 6: Stereo Matching

Charlotte Moraldo

Note: As the included package "sift" wasn't working on my computer, I used the VLFeat toolbox from http://www.vlfeat.org

1. Disparity computation

I begin by rectifying the two given images and by computing an average filter of a certain window size. I shift the entire image 2 in the disparity range, and compute the squared difference (SSD) between image 1 and the shifted image 2. The result is convoluted with the average filter. For each pixel, I then remember the best disparity.
The following images show the disparity maps obtained with different window sizes:

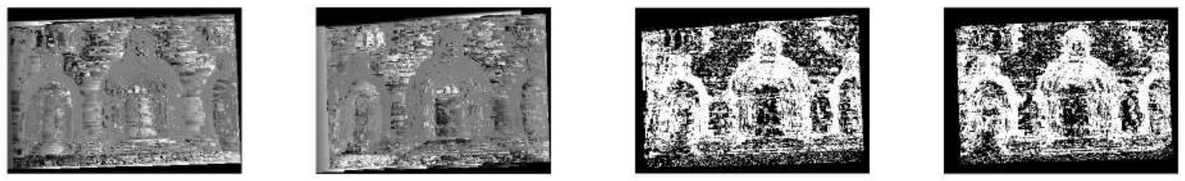

Figure 1: Disparity maps with filter window size: 3x3

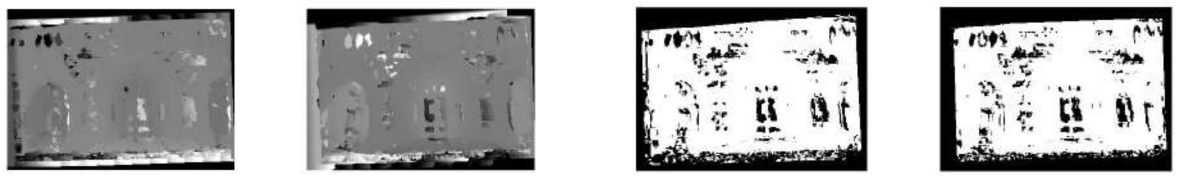

Figure 2: Disparity maps with filter window size: 10x10

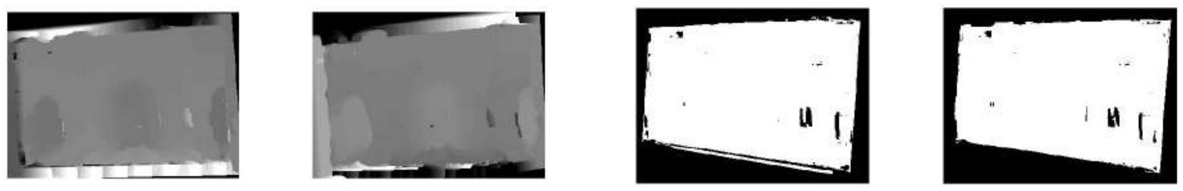

Figure 3: Disparity maps with filter window size: 20x20

We can see that as with a small filter window size, there are a lot of depth discontinuities. By increasing it, the disparity map becomes more and more smooth. With a window size of 20x20, all the depth discontinuities are gone, and the result is very smooth.

2. Graph-cut

In this implementation of stereo matching, I model the disparity computation as a graph labeling problem. Each pixel corresponds to a node, and each disparity to a label. The goal is to assign edge weights so that the min-weight cut gives the minimum energy. Therefore, matching pixels have similar intensities and most nearby pixels have similar disparities.
The following images show the disparity maps obtained with different window sizes:

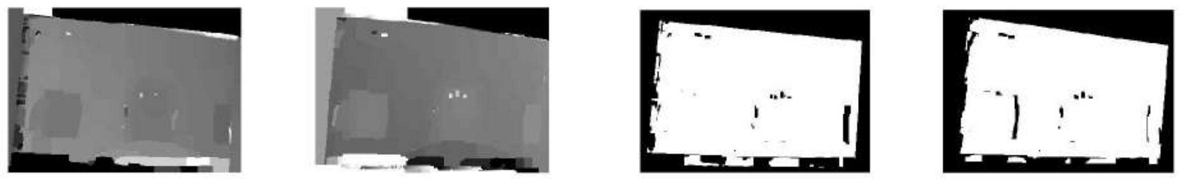

Figure 4: Disparity maps with filter window size: 3x3

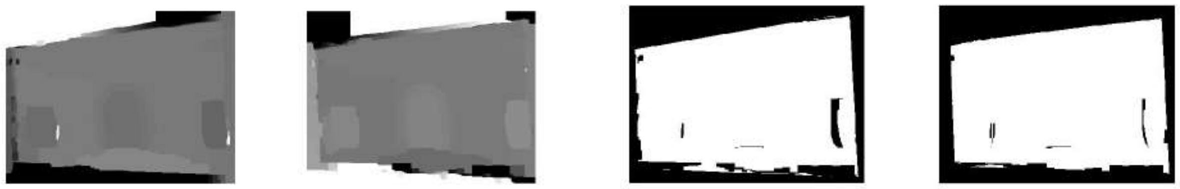

Figure 5: Disparity maps with filter window size: 10x10

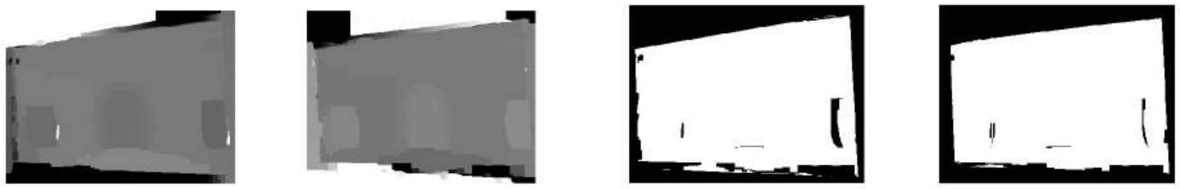

Figure 6: Disparity maps with filter window size: 20x20

The results obtained with graph-cut are much better than those computed in part 1. Indeed, we can see for example that for a filter size of 3x3, the depth discontinuities are already almost gone with graph-cut, while they were plenty left in part 1. As we further increase the window size, the disparity maps don't even change anymore since they are already very smooth, and all the discontinuities are gone.

3. Generating a textured 3D model

For each pixel, I compute the corresponding 3D point, using the projection matrices of both images and the disparities calculated in parts 1 and 2. This is mostly done thanks to the matlab function `linTriang`, which triangulates two points $x1$, $x2$, using the corresponding projection matrices. Therefore, for each pixel, the 3D coordinate is given by:

```
3Dcoord = linTriang([x y], [x-disp y], PL, PR)
```

Where `[x y]` is the coordinate of the current pixel, `disp` is the disparity, `PL` and `PR` are the projection matrices of both images. The following images show the results obtained for both methods, with different filter sizes.

3.1 Disparity computation

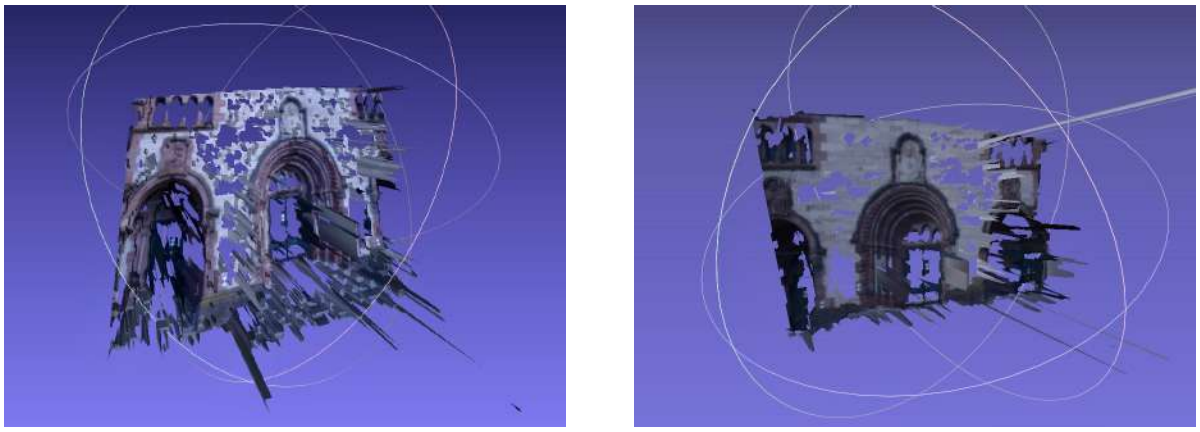

Figure 7: Front and back view of the 3D reconstruction, with filter window size 10x10

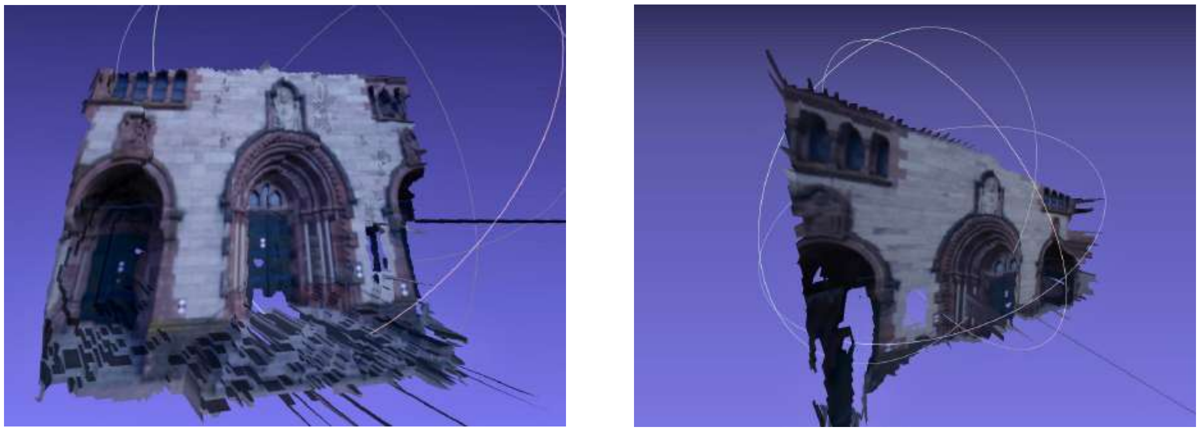

Figure 8: Front and back view of the 3D reconstruction, with filter window size 20x20

As what was seen with the disparity maps obtained, the results with a larger window size (20x20) are much better. The disparity map was smoother, and now we see that the 3D reconstruction is much better than with a smaller window size. In the case where we take a 10x10 sized filter, the depth discontinuities can be seen here, as the reconstruction has many ”gaps” and lacks information towards the windows, the doors, and the front stairs.

3.2 Graph-Cut

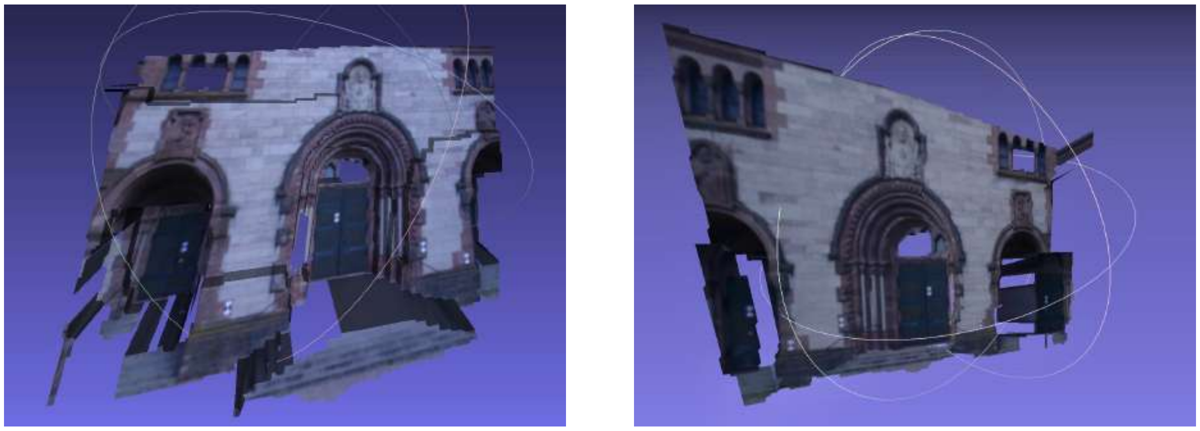

Figure 9: Front and back view of the 3D reconstruction, with filter window size 3x3

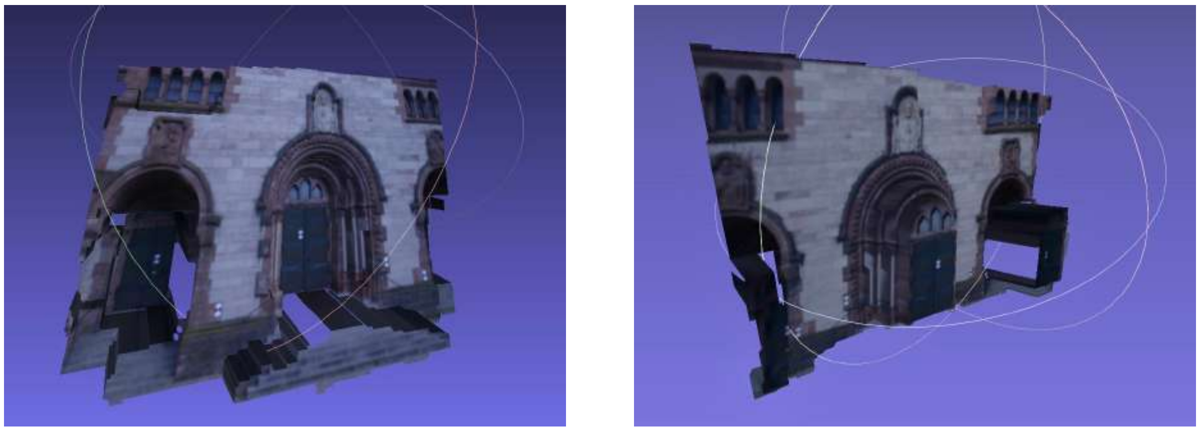

Figure 10: Front and back view of the 3D reconstruction, with filter window size 10x10

The results obtained with graph-cut are very good. Even with a very small window size of 3x3, the 3D is as good as what was obtained previously with a 20x20 window: for example, the front stairs are much better reconstructed and very few ”gaps” can be seen. The windows and the doors are the main areas who still need improvement.

The result obtained with a 10x10 sized filter is the best reconstruction obtained. The only remaining gaps are those caused by a flat areas who change in depth (i.e. large wall or floor areas going towards the ”inside” of the picture). It makes sense that the reconstruction of such parts is more difficult: since one same area varies a lot in depth, the algorithm doesn’t know how to label it and which disparity to give. Despite this, this final reconstruction already gives a very good insight of what the real 3D landscape looks like.