# Computer Vision

# Assignment 6: Stereo matching

*Student: Ivan Alberico*

## 1. Disparity computation

The first thing to do before computing the disparities is rectifying the two images, which brings the epipoles at infinity and makes the epipolar lines parallel to each other. In this way, disparity can be safely computed as a shift in the horizontal direction. In order to implement the winner-takes-all algorithm, given the disparity range, I first shift the second image for every value in this range and then I compute the SSD between the shifted image and the non-shifted one. The results are then convoluted with an average filter, and finally I only take the best disparities, such as the ones with the smallest SSD.

The size of the filter window is a very important parameter to choose, since it affects how the disparity man is sensitive to noise. The larger the window size is, the less the disparity map would be affected by noise in the flat regions, resulting in a smoother outcome. On the other hand, choosing a very big window might lead to some minor characteristics being overlooked (since we are averaging on a wider set of values) and it might happen that some important information gets lost. Different values of the window size were tested, and the results obtained are the following:

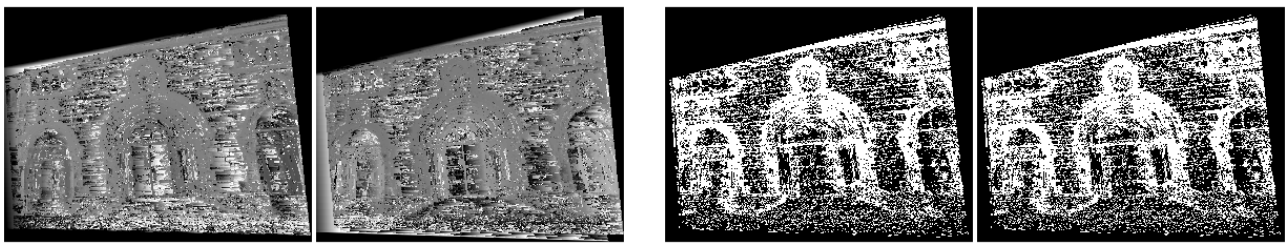

**Figure 1** — Disparity map computed with the winner-takes-all algorithm (with SSD) and using a window size of 3x3.

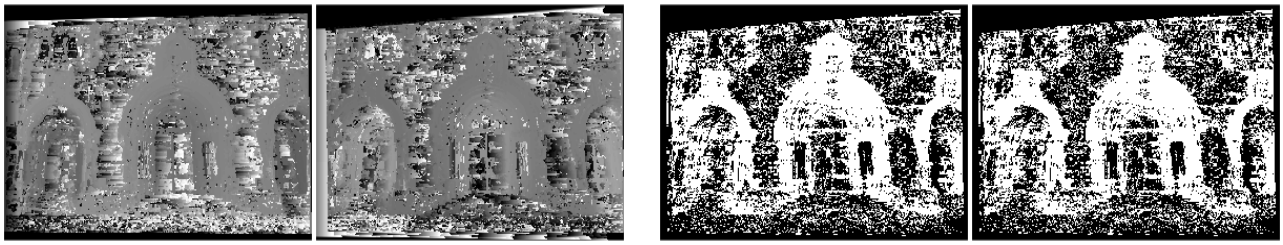

**Figure 2** — Disparity map computed with the winner-takes-all algorithm (with SSD) and using a window size of 5x5.

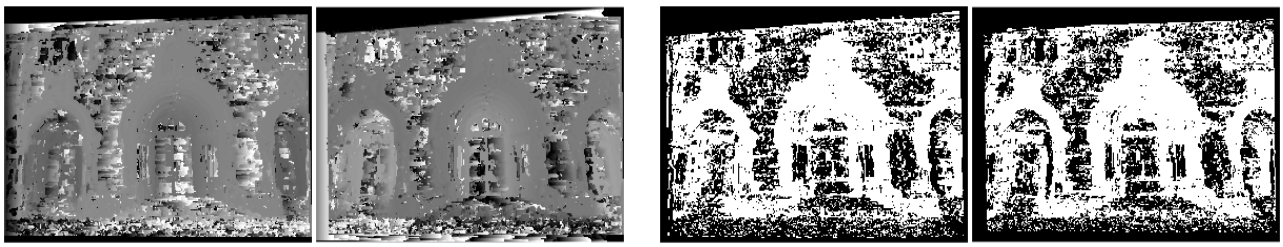

**Figure 3** — Disparity map computed with the winner-takes-all algorithm (with SSD) and using a window size of 7x7.

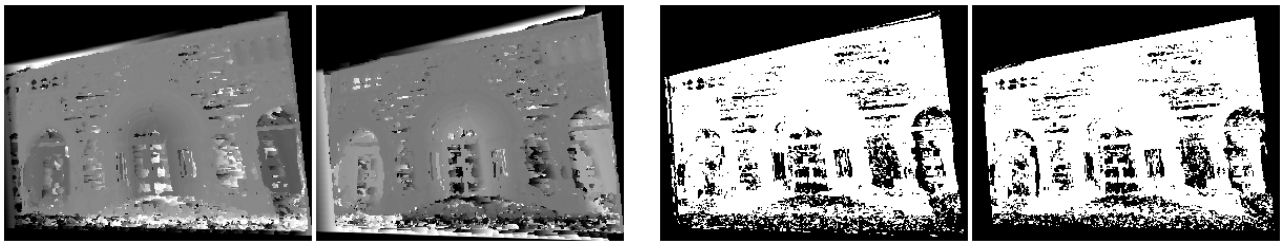

**Figure 4** — Disparity map computed with the winner-takes-all algorithm (with SSD) and using a window size of 10x10.

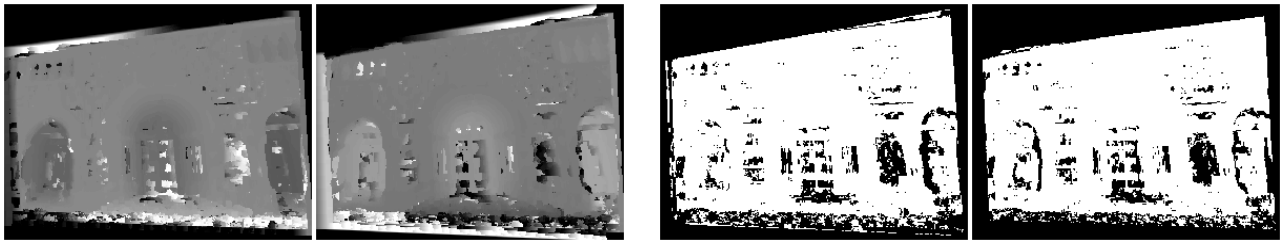

**Figure 5** — Disparity map computed with the winner-takes-all algorithm (with SSD) and using a window size of 15x15.

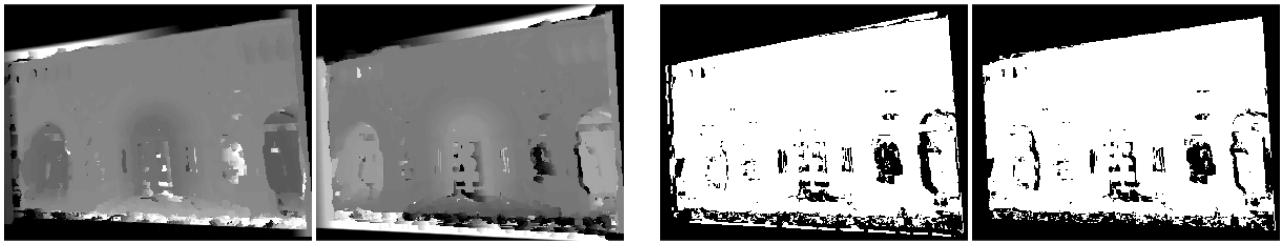

**Figure 6** — Disparity map computed with the winner-takes-all algorithm (with SSD) and using a window size of 20x20.

From the previous images, it is evident that good results are not achieved for small sizes of the filter window. In fact, the disparity maps show many depth discontinuities, even between points that are very close to each other and that should instead present a more continuous behaviour (they should have similar 3D depth). As the window size increases, these depth discontinuities diminish and smoother areas are obtained.

Very similar results are obtained by computing the distances using SAD instead of SSD, as shown below:

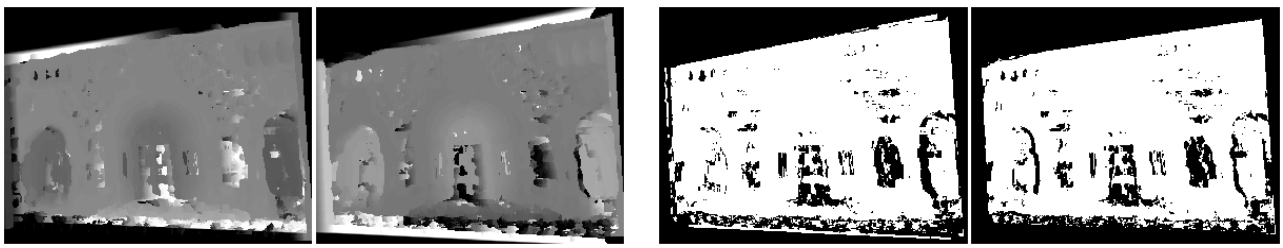

**Figure 7** — Disparity map computed with the winner-takes-all algorithm (with SAD) and using a window size of 20x20.

## 2. Graph-cut

The Graph-cut algorithm, instead, computes the disparity as a graph labelling problem, where each pixel corresponds to a node and each disparity to a label. What we need to do is to define a cost function and minimize over the labels. The cost function we take into account is made of two terms: the first one is the cost of assigning a label to each pixel, while the second is related to assigning labels to the neighbouring pixels (hence, regulating the smoothness).

While the smoothness term is already given, the data cost $D_C$ is obtained by computing the disparity for each pixel, by using the SSD. By tuning the weighting parameters of these two cost function terms, as well as the size of the average filter window in the data cost term and the size and the sigma of the Gaussian filter (used to compute the vertical and horizontal gradients for the Graph-cut), I was able to improve the performance and obtain very good results with this method.

Let us first see the effects of just changing the size of the average filter window:

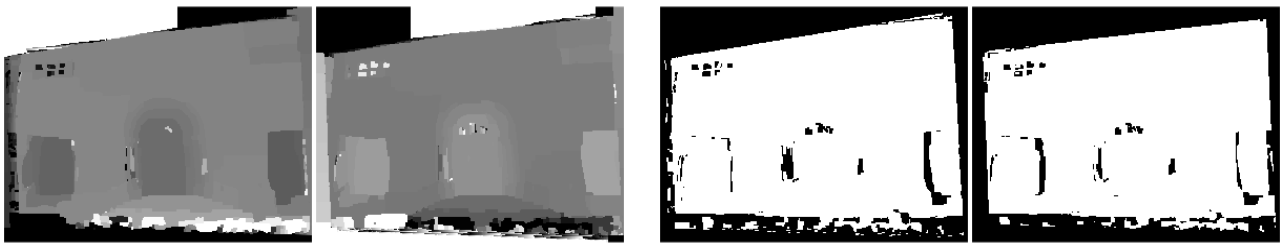

**Figure 8** — Disparity map computed with the Graph-cut algorithm and using a window size of 3x3.

**Figure 9** — Disparity map computed with the Graph-cut algorithm and using a window size of 5x5.

**Figure 10** — Disparity map computed with the Graph-cut algorithm and using a window size of 10x10.

**Figure 11** — Disparity map computed with the Graph-cut algorithm and using a window size of 20x20.

Overall, the Graph-cut method clearly outperforms the winner-takes-all algorithm. It is evident from the previous images that, even by choosing a small filter size, say 3x3, the Graph-cut method already performs much better, showing a smooth disparity map and very few depth discontinuities.

As a drawback, this algorithm is much more computationally expensive than the other. By keeping the scaling factor fix to $0.5^2$, it takes around 5-6 minutes to the Graph-cut algorithm to execute, which is roughly 8-10 times the time needed to the WTA to run.

Once the disparity map has been computed, in order to obtain the 3D model, the points correspondences between the two images needed to be triangulated. The textured object file is then created and it is visualized through the MeshLab software.

The 3D models obtained with the winner-takes-all algorithm, respectively with a window size of 5x5 and 20x20, are the following:

**Figure 12** — Textured 3D model obtained with the winner-takes-all algorithm, with a window size of 5x5 (on the left) and 20x20 (on the right).

The results obtained from the 3D representations are compliant with what was discussed before. In the case of the 5x5 window, we can see that the model shows so many depth discontinuities that is quite

difficult to actually recognize the real image. By using a 20x20 window, we do have a better model, however we still face some irregularities and holes which are not encountered with the Graph-cut model.

The 3D model obtained with the Graph-cut method, which will be now shown, has been computed by setting the following values for the hyperparameters: size of the average filter window = 10x10, size of the Gaussian filter window = 3x3, sigma of the Gaussian filter = 5, weighting term of the data cost = 800, weighting term of the smoothness cost = 5 (these last two terms are actually very important since they allow us to regulate the trade-off between smoothness and continuity of the labels in our model).

**Figure 13** — Textured 3D model obtained with the Graph-cut algorithm.

From the model obtained, it is clearly visible that the Graph-cut algorithm performs pretty well. The model is indeed very smooth, presenting little or even no depth discontinuities, and it provides a trustworthy reconstruction of the scene.

## 3. Automate disparity range

So far, we have computed our results based on a fixed disparity range (namely -40:40), which was chosen a priori (given by the problem description). In practical applications, instead, it is very important that this value is set automatically, so that the algorithm is not hardcoded to some prior assumption but is able to get this value independently.

Before exploring how the disparity range can be automatically computed, it would be useful to first estimate the disparity range by manually clicking a set of correspondences between the two images, having different depths. In this way, one could start having a rough idea on how the optimal disparity range would look like for this specific task, so that we can later compare if our automatic algorithm is actually providing reasonable results or not.

Once the correspondences are selected, the disparity range is calculated by considering the max and min values among these. This is implemented in the function `setRange_clickPoints.m` which uses the function `getClickedPoints.m` to obtain the points correspondances. An example is shown below:

**Figure 14** — 6 points' correspondences manually selected with `getClickedPoints.m` function, which give a disparity range of -8:8.

In *Figure 14*, the correspondences in both the two images have been selected by clicking points with different depths. The 6 selected correspondences produce a disparity range of -8:8. By selecting other correspondences, similar ranges were obtained, going from -7:7 to -10:10.

In order to automatically calculate the disparity range, the first thing that was done was extracting the SIFT features of the two images and matching them. The disparity range was computed by taking the difference between the obtained features in both images, then computing the max and min value of this difference and finally considering the max absolute value between the two, as the range size.

However, by just following this procedure, results were not always very accurate. A proper filtering of the outliers was needed, and this was achieved by means of RANSAC adaptive algorithm. After applying RANSAC, in fact, results were more stable and accurate.

Some hyperparameters were tuned to achieve good results: 'PeakThresh' was set to 0.001, 'WindowSize' to 7 (these two values are the parameters of the `vl_sift.m` function), the threshold for the feature matches was set to 2 (in the `vl_ubcmatch.m` function), while the threshold for the RANSAC algorithm was set to 3 (I used the function `ransac8pF.m` which I implemented in the Assignment 4).

By running the algorithm several times, I obtained an average disparity range size of -8:8 (overall, the range size varies between -6:6 and -12:12), which is compliant with the results I obtained before by directly clicking points' correspondences on the two images.

However, the algorithm was yet not completely robust, since sometimes I was still obtaining some irregular range sizes, for example of -65:65 (this was happening once every 20-30 times I ran the algorithm, so it was probably caused by some minor outlier which was not correctly filtered out). To overcome this issue, instead of considering the max and min value to compute the disparity range, the 1$^{st}$ and 99$^{th}$ percentile were set as the lower and upper bound for our calculation. In this way, the few outliers still left were not taken into account anymore.

This was all done in the `automate_disparity_range.m` function I implemented on Matlab.

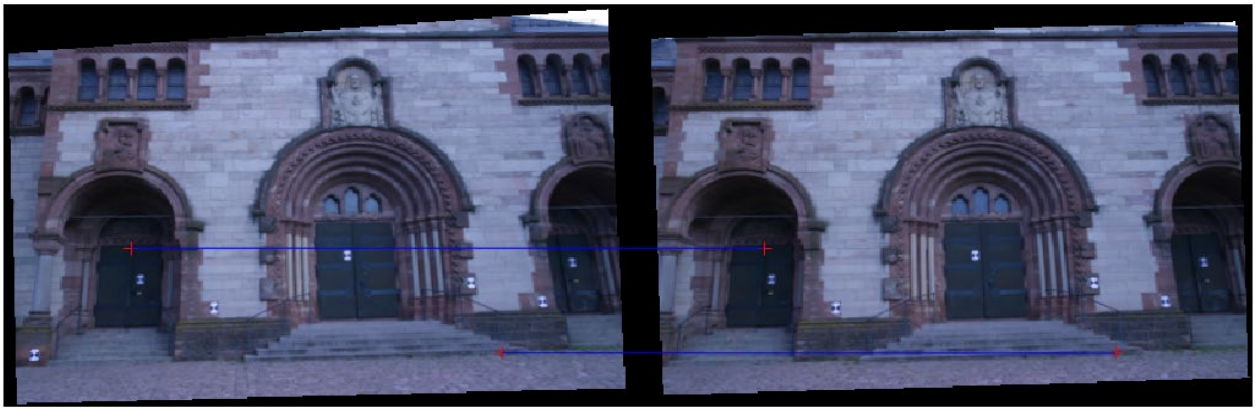

**Figure 15** — Feature matches of the points corresponding to the max and min disparity values, leading to a disparity range of -8:8.

**Figure 16** — Feature matches of the points corresponding to the max and min disparity values, leading to a disparity range of -9:9.