Computer Vision Homework n.2 Report

DISPARITY COMPUTATION

June 24, 2019

Claudio Curti ID: 4216203
Nicola De Carli ID: 4198668
Alberto Ghiotto ID: 4225586
University of Genoa

Contents

Chapter 1

Function examination

1.1 my_ssd

This function simply computes the "sum of squared differences" (SSD) with a minus sign (the minus is simply because we want to use this function as a similarity function), given as input the patches belonging to a stereo pair of rectified images.

1.2 my_disparity

This function is the fulcrum of this laboratory as it computes the disparity map between the images. It takes as input, in addition to the two images, also the dimension of the window W, on which the SSD will be computed, and the range [dmin, dmax], specific for each pair of images, that allows to make the search of the best correspondence more efficient. In this function the goal is to find correspondent pixels between the two images to estimate their disparity, and therefore to have a sort of measure of the relative distance between the different elements represented in the two images (in the 3D world) with respect to the origins of the frames of the two images.

To do this, for each disparity d in the search range found for each pixel pl of (i,j) coordinates in Il, the similarity between pl and each pixel pr of (i,j+d) coordinates is estimated using the my_ssd function, to find the d value corresponding to the most similar pixel (that with the highest value of ssd, here it is the reason of sign minus).

The value of d obtained in this way is saved in a matrix D in the coordinates (i,j). The function gives back the matrix D containing the distance d in pixel of each corresponding pixel in Ir with respect to Il.

1.3 my_disparity_final

In addition to the two images this function also takes as input the dimension of the window W, on which the SSD will be computed, and the range $[dmin, dmax]$, specific for each pair of images, that allows to make the search of the best correspondence more efficient.

This function simply calls the function "my_disparity" initially to compute the disparity from left to right and then to compute the disparity from right to left. In this second case, some precaution has to be applied, not only the order of input images to "my_disparity" have to be changed but also the argument passed to $dmin$ becomes $-dmax$ and that for $dmax$ becomes $-dmin$. Eventually, a consistency check is performed, which consists in checking that $Dlr(i,j) = -Drl(i, j + Dlr(i,j))$, where Dlr represents the disparity map from left to right and Drl from right to left. In this way, a consistency matrix of the same size as Dlr and Drl is filled with 1 and 0 based on the fact that for a given element of the matrix the consistency check is passed or not.

Chapter 2

Implementation and results

2.1 Implementation

As suggested, the "script_disparity.m" MATLAB script was a bit modified in order to check the behaviour of the current implementation.
The script loads the left and right images from memory and perform the rectification, to simplify the problem of finding matching points between images.

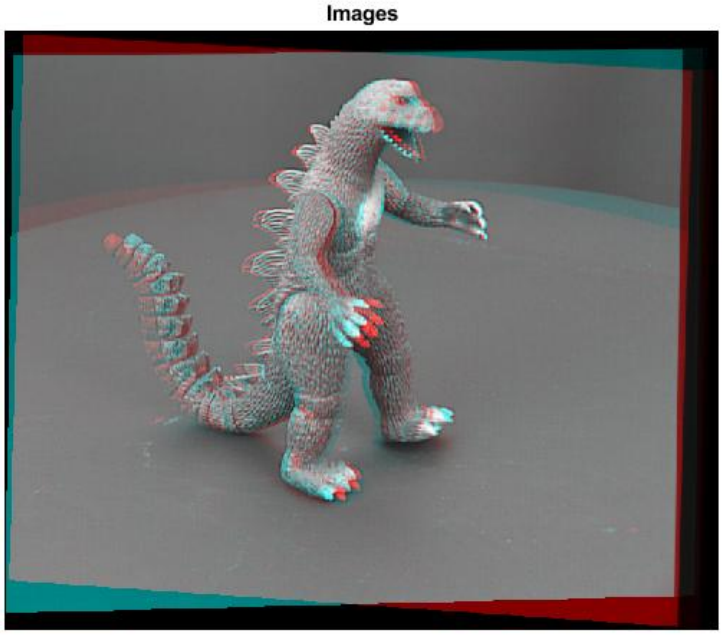

Figure 2.1: Rectified dinosaur image

Figure 2.2: Left dinosaur images

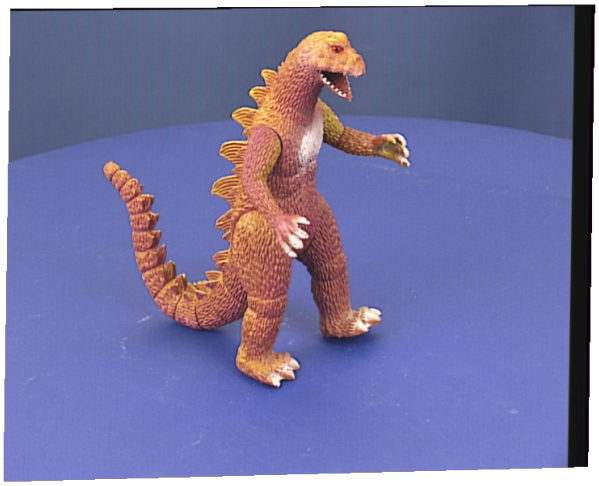

Figure 2.3: Right dinosaur image

2.2 Conclusions

It has been noticed that, particularly in the uniform areas, the disparity map tends to get wrong, as could have been expected since the similarity function in those areas provide no relevant information. Furthermore, it has been noticed that using the algorithm implemented in this way the computational time grows a lot when increasing the size of the images, it is particularly evident comparing the case of the images of the dinosaur, which are very big with respect to the others, with the others.

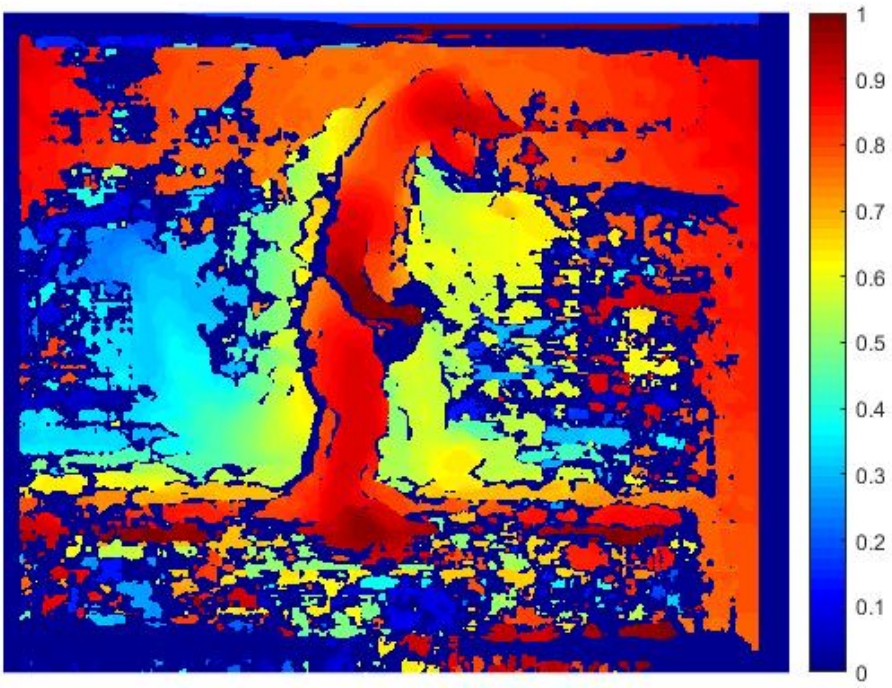

Figure 2.4: Disparity map of dinosaur image computed with MATLAB "disparity" function

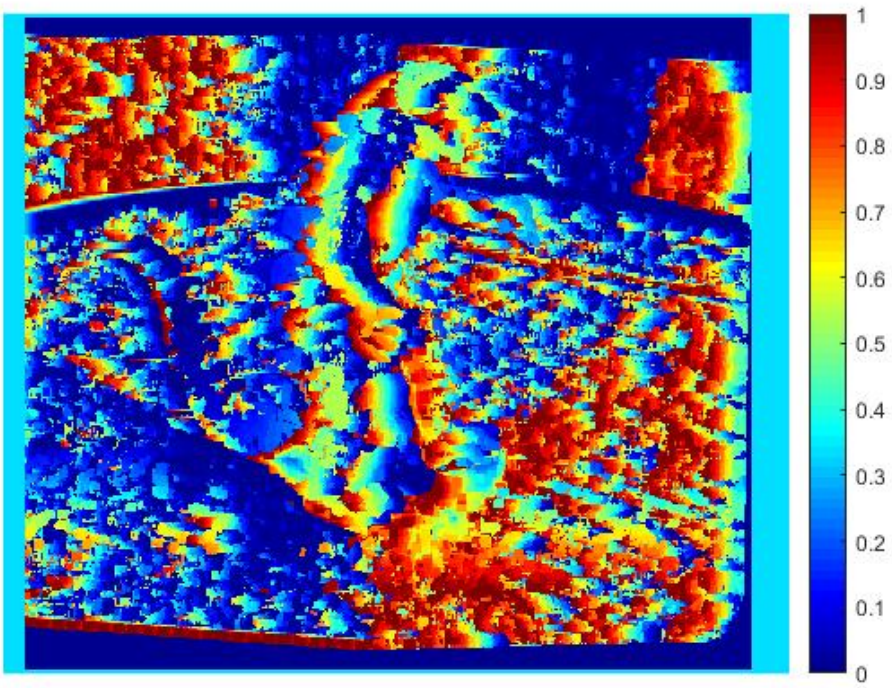

Figure 2.5: Disparity map of dinosaur image computed with "my_disparity_final" function

As visible in the images below, the disparity map is also useful to detect some "hidden" pattern that would not be noticed in the normal rectified image.

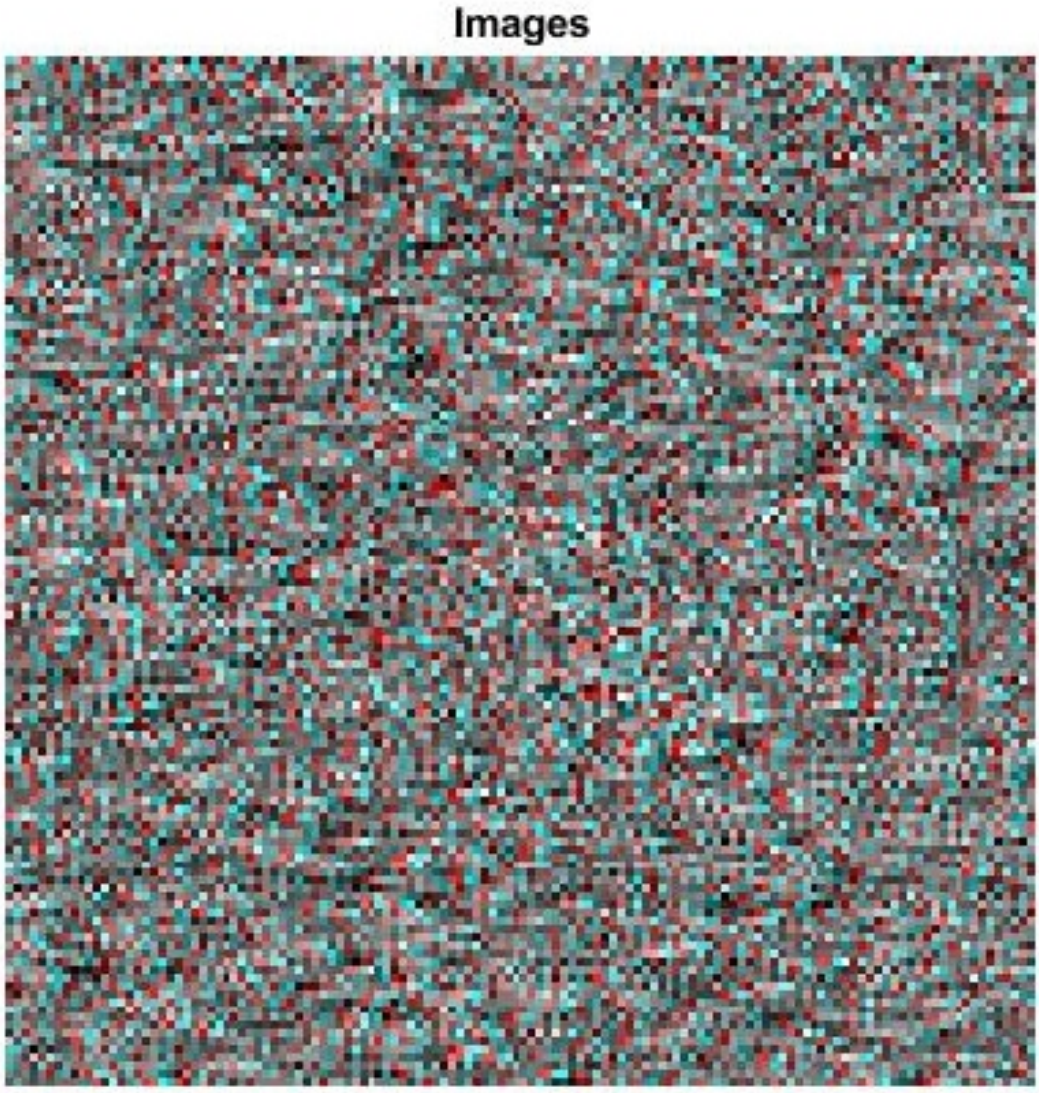

Figure 2.6: Rectified "white noise" image

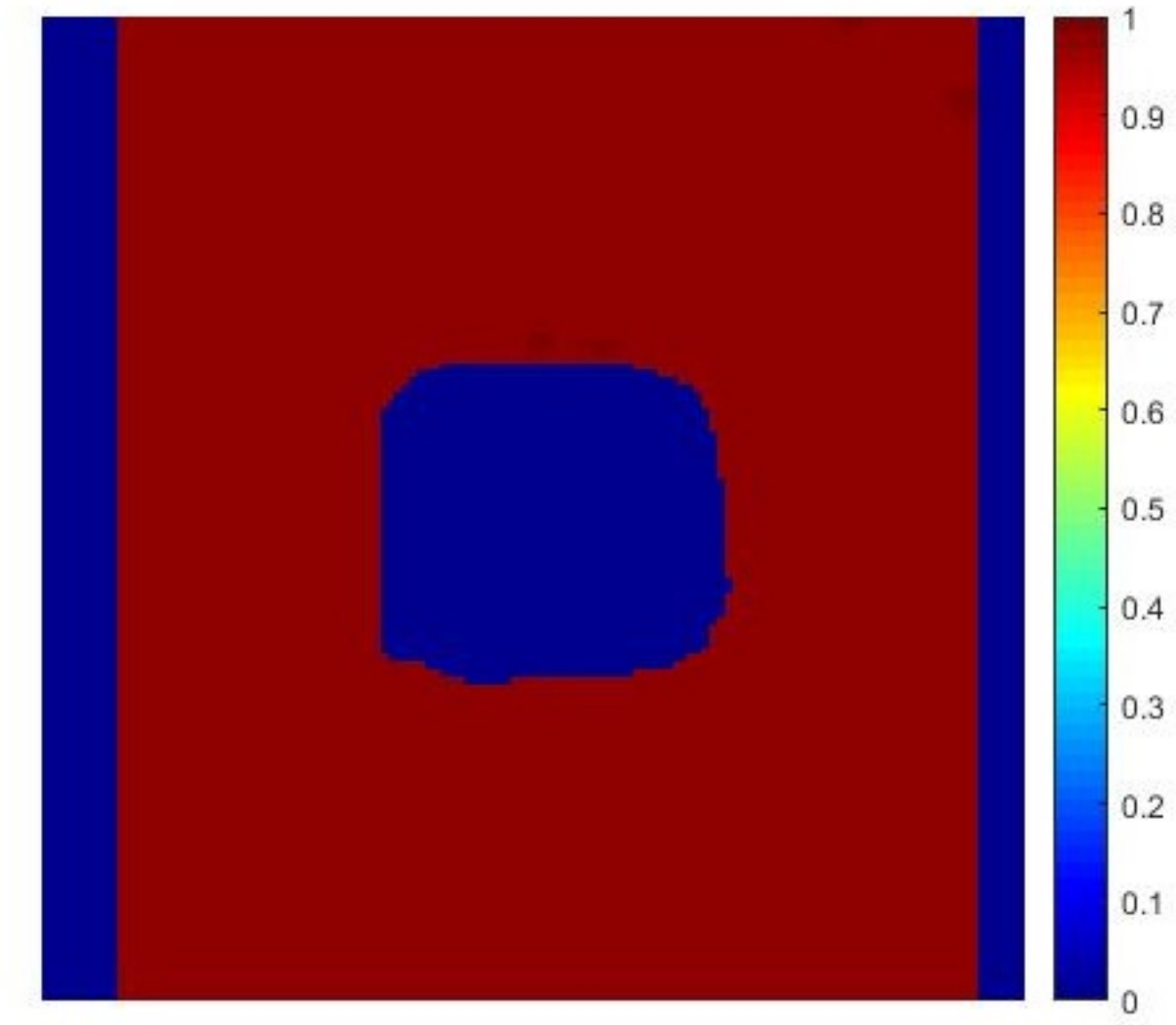

Figure 2.7: Disparity map of "white noise" image computed with MATLAB "disparity" function

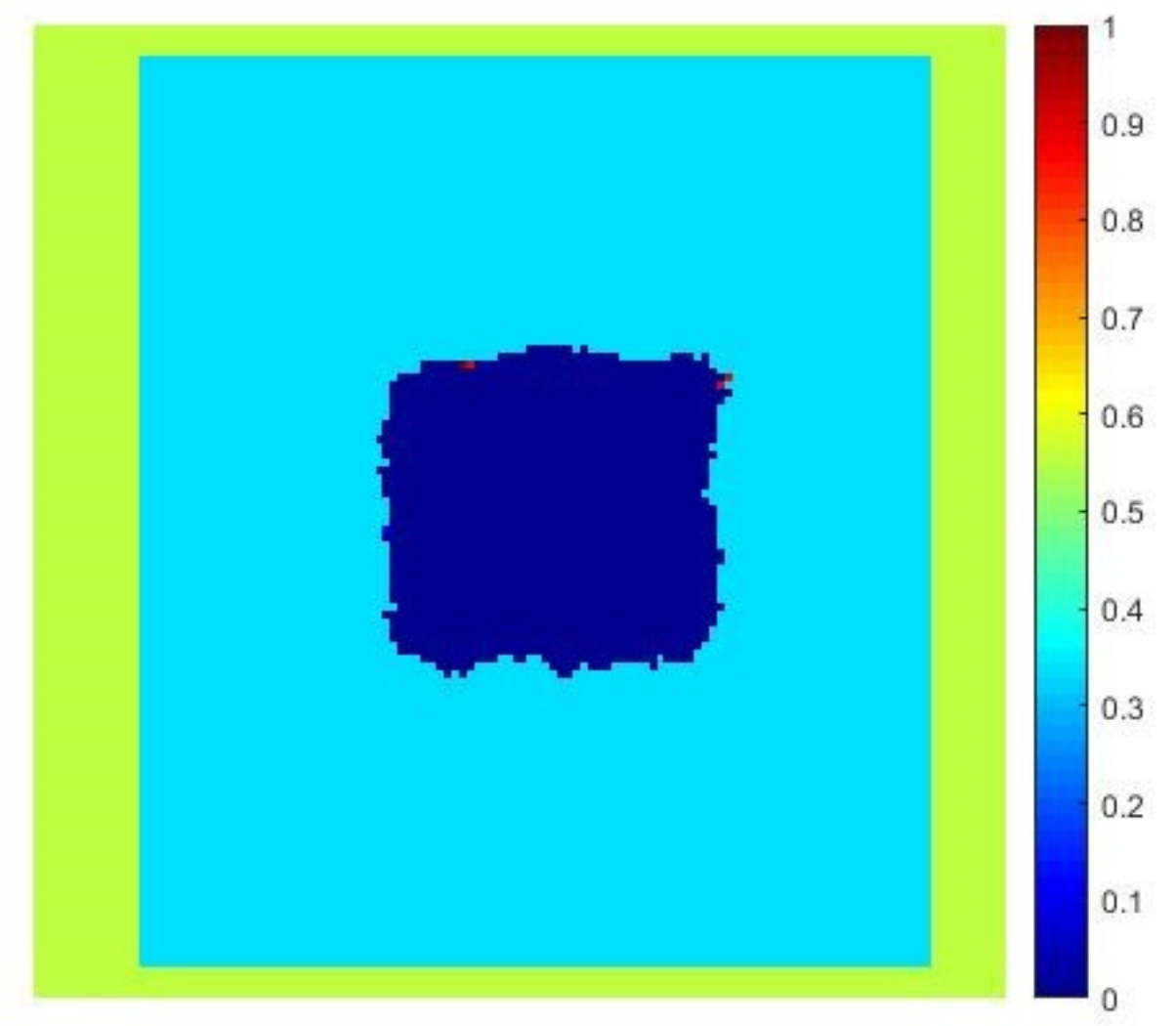

Figure 2.8: Disparity map of "white noise" image computed with "my_disparity_final" function