Module 4 - Report week 4

Hicham El Muhandiz[1[0000−1111−2222−3333]] , Julia Ariadna Blanco Arnaus[1[0000−1111−2222−3333]] , Marcos Muñoz González[1[0000−1111−2222−3333]] , Rosana Valero Martínez[1[0000−1111−2222−3333]]

Universitat Autònoma de Barcelona, Campus UAB, Edifici O, 08193 Cerdanyola del Vallès, Barcelona

1 Introduction

In this project, the matching correspondences between two views of the same scene are triangulated by applying the Direct Linear Method (DLT). Then, camera matrices of these images are computed by using the Fundamental Matrix and the Intrinsic Matrix, and the triangulation method is evaluated on these camera matrices by the estimation of the reprojection error in the triangulation. Later, depth maps are computed using local methods (SSD, NCC) and implementing bilateral weights on the mapping.

2 Implementation

2.1 Triangulation with the DLT method

Given a set of matching points x and x' as well as the camera matrices P and P', we want to compute the 3D points X corresponding to the matches of the two images such that,

$$\mathbf{x} \equiv P\mathbf{X} \qquad\qquad \mathbf{x'} \equiv P'\mathbf{X}$$

where

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \mathbf{x'} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} P = \begin{pmatrix} \mathbf{p}^{1T} \\ \mathbf{p}^{2T} \\ \mathbf{p}^{3T} \end{pmatrix} P' = \begin{pmatrix} \mathbf{p'}^{1T} \\ \mathbf{p'}^{2T} \\ \mathbf{p'}^{3T} \end{pmatrix} \tag{1}$$

This problem can be written as:

$$\mathbf{AX} = 0 \tag{2}$$

where

$$\mathbf{A} = \begin{pmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p'}^{3T} - \mathbf{p'}^{1T} \\ y'\mathbf{p'}^{3T} - \mathbf{p'}^{2T} \end{pmatrix} \tag{3}$$

The solution to this can be found using the Triangulation with the DLT (Direct Linear Transformation) method by minimizing $\|\mathbf{AX}\|^2$ subject to $\|\mathbf{X}\|^2 = 1$.

To do so, we build the Homography matrix H, taking into account the normalization according to the scaling and translation so that both pixel coordinates are in the interval [-1, 1]:

$$H = \begin{pmatrix} 2/nx & 0 & -1 \\ 0 & 2/ny & -1 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

Then, we build the matrix A by multipliying the points and camera matrices with the Homography matrix, as follows:

$$x \equiv HX \qquad x' \equiv Hx' \qquad P \equiv HP \qquad P' \equiv HP'$$

Once we have done this, for each image point, we create the matrix A (using Equation (3)) and use the Singular Value Decomposition (SVD) to calculate the homogenous coordinates of the X points in 3D space.

To evaluate our implementation of the triangulation function, we have to calculate the average triangulation error and obtain a value lower than 10^{-8}. As we have obtained an error of 1.50^{-15} [m], we can consider it correct.

In Figure 1 we can observe that the triangulated values are close to the ground truth.

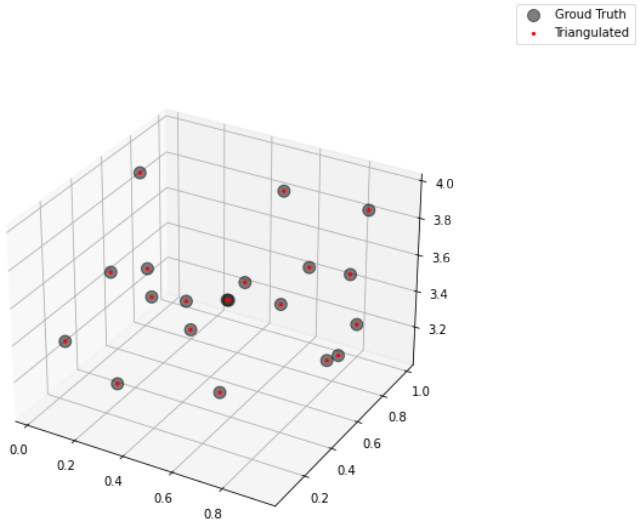

Fig. 1: Plot of the results of the triangulation with the triangulated and groundtruth values.

2.2 Reconstruction from two views

In this section the goal is to estimate the 3D reconstruction from two views in a situation where the image correspondences contain outliers. First, a set of point correspondences between two images is searched (we use gray-scaled facade images). ORB is used to find the image keypoints and descriptors and a brute-force descriptor matcher retrieves the matches between two images.
As this correspondences contain noise/outliers, the 8-point algorithm is used to find the fundamental matrix F. However, using only 8 correspondances may lead to bad estimates of F, so in order to increase the robustness of the estimation the RANSAC algorithm is used.
To compute the Essential Matrix from the Fundamental Matrix, the following equation is used:

$$E = K'^{T} F K \tag{5}$$

Once the essential matrix is known, the camera matrices can be retrieved from E up to scale and a four-fold ambiguity. If we assume that the first camera matrix is $P = [I|0]$, in order to compute the second camera matrix $P^{'}$ is necessary to factor E into the product SR of a skew-symmetric matrix and a rotation matrix. If we suppose that the SVD of E is $Udiag(1, 1, 0)V^T$ and we use the following notation:

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{6}$$

There are two possible factorizations: $S = UZU^T$ and $R = UWV^T$ or UW^TV^T. From $S = [t]_x$ we can determine the translation part of the camera. Since $St = 0$, it follows that:

$$t = U \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T = u_3 \tag{7}$$

However, the sign of E, and consequently t, cannot be determined. Because of this, we have 4 possible camera matrices based on the two choices of R and the two possible signs:

$$P^{'} = \left(UWV^T| + u_3\right) P^{'} = \left(UWV^T| - u_3\right) P^{'} = \left(UW^TV^T| + u_3\right) P^{'} = \left(UW^TV^T| - u_3\right) \tag{8}$$

Testing with a single point is enough to see if is in front of both cameras (the canonical and one of the 4 possible solutions) is sufficient to decide between the four possible solutions for the second camera matrix P'.

In Figure 2 we observe the different cameras obtaind and how they are located in relation to the canonical camera. In Figure 3 it can be observed how they are located in regards to the other proposed cameras as well as their different rotation.

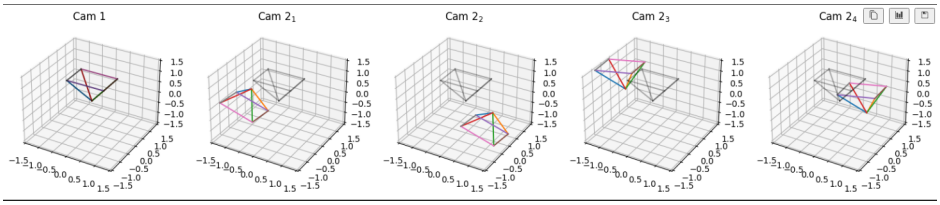

Fig. 2: Plot of the different cameras with the canonical camera

To test if point X = (x, y, z, 1) is in front of the camera, we compute $X' = P * X$. $X' = (x', y', z')$ will be the position of the point if the camera is in the origin, looking toward z direction. So, if z' is positive, then X' is in front of the camera and X is in front of camera P. In Figure 4 we can observe how the third camera is the one which meets this condition, and therefore, the only possible solution.

After selecting the optimal camera, we can visualize in Figure 5 the 3D points with an interactive plot to see the surfaces reconstructed in 3D as well as the image with the selected points.

The 3D reconstruction can be analyzed also by looking at the error obtained when reprojecting the 3D space back to image coordinates. The reprojection error is a geometric error corresponding

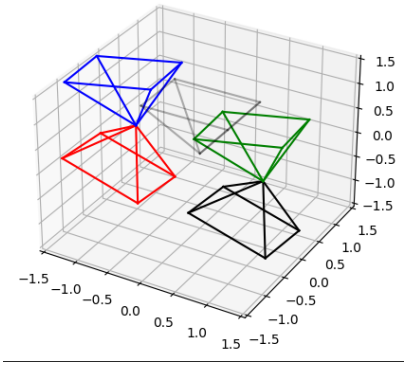

Fig. 3: Plot of the different cameras with the canonical camera in the same image

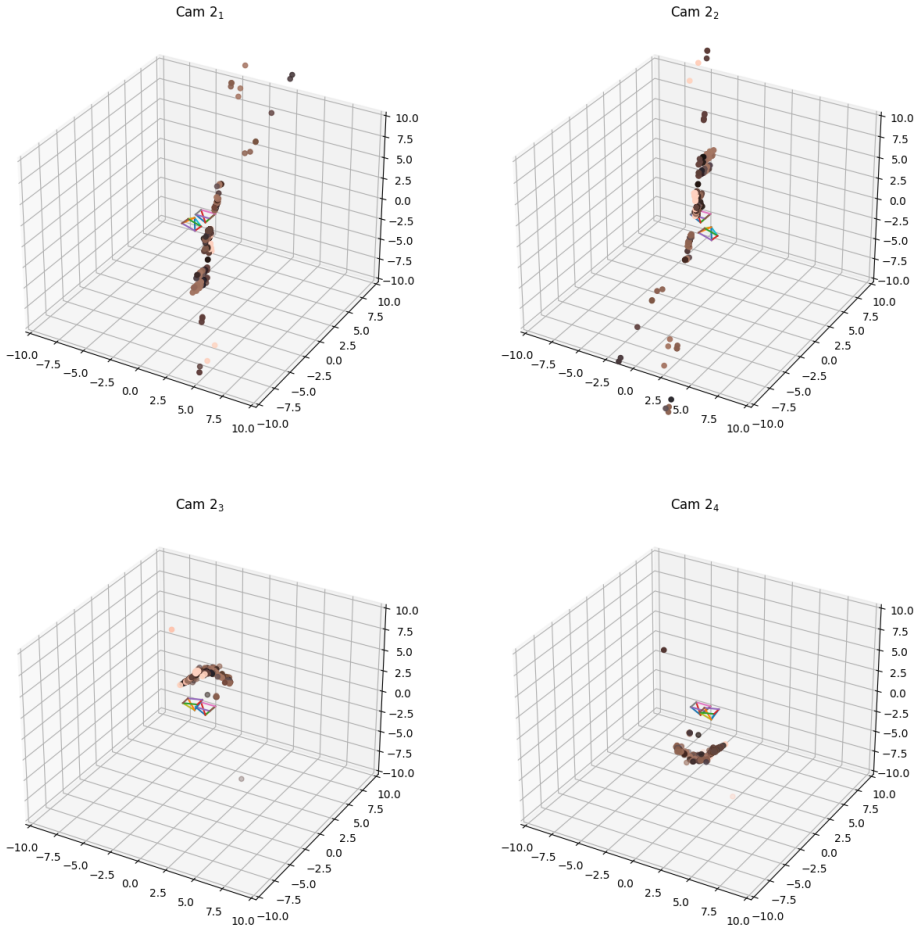

Fig. 4: Points plotted to see if they are in front of both cameras

Fig. 5: Visualization of the 3D points for the selected camera and image with the selected points

to the image distance between a projected point and a measured one. The following equation is used to compute the reprojection error:

$$\sum_{i=1} d(x_i, \hat{x}_i)^2 + d(x_i^{'}, \hat{x}_i^{'})^2 \tag{9}$$

In Figure 6, we observe the histogram obtained for the reprojection error in both cameras. We can see that columns have a mean error between 0 and 10 pixels.

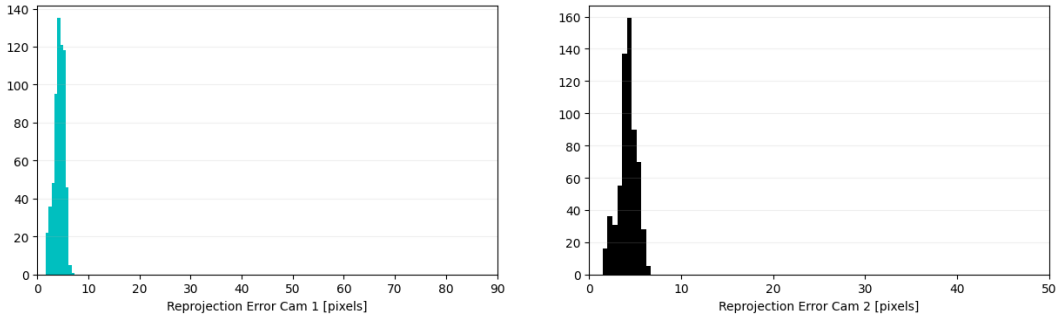

Fig. 6

2.3 Depth map computation using local methods

Given the differences between the pixels of two stereo images, we can get an estimate of the depth of a scene by computing the disparity map. The idea behind this algorithm is to locally compare the pixels of one of the images to different windows near that position in the other image and find the window that minimises a cost function.

Disparity between a pair of rectified images To compute the disparity map we slide a window in each row of the first image and compute the cost function in the neighbourhood of that position of the second image. A way to define the cost function is shown in Equation 10, where p is the pixel, d is the disparity, I_1 and I_2 are the stereo images, and w defines a set of weights. If we use m=1, we will be using as a matching criteria the Sum of absolute differences (SAD). After computing the cost function for all the nearby windows on the second images, the disparity of the pixel that minimises the cost C will be the value of the disparity map. A consideration we had to take into account when implementing this was how to deal with the scenario where the search windows are out of bounds of the second image. In our case, we do not take into account these situations when computing the disparity map.

$$C(p,d) = \sum_{q \in N_p} w(p,q)|I_1(q) - I_2(q+d)|^m \tag{10}$$

In Figure 7, we can see some results using different window sizes. In particular, using smaller windows results in a noisy result with lots of sharp edges, whereas increasing the size of it grants smoother contours and a less grainy outcome. There is a trade-off between the amount of details we want to keep and how noisy will our representation be.

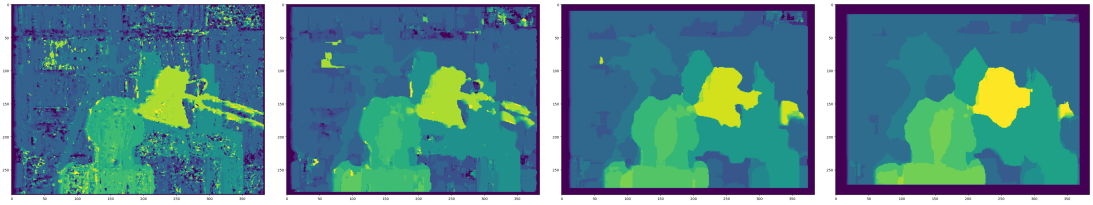

Fig. 7: Output using SAD and window sizes 3x3, 9x9, 21x21, 31x31

Sum of Squared Differences Cost A different way to obtain the disparity map is to tweak the cost function to compute the Sum of squared differences (SSD) instead of the SAD. This can be done by changing m to 2 in Equation 10.

We can obtain different results by modifying the window size (Figure 8). Similarly to what happened using SAD, increasing the window smooths out the disparity map. When using smaller window sizes, there is not much difference between using SAD or SSD. However, as we increase the window size, the regions or depth levels output by SAD are more homogeneous than the ones output by SSD.

Normalized Cross Correlation Cost Another modification is to switch the previous cost function by the normalized cross correlation cost (Equation 11), where $\overline{I}_1$ and $\overline{I}_1$ are the means of each image and $sigma_{I_1}$, $sigma_{I_1}$ the standard deviation.

$$NCC(p,d) = \frac{\sum_{q \in N_p} w(p,q)(I_1(q) - \overline{I}_1)(I_2(q+d) - \overline{I}_2)}{\sigma_{I_1}\sigma_{I_2}} \tag{11}$$

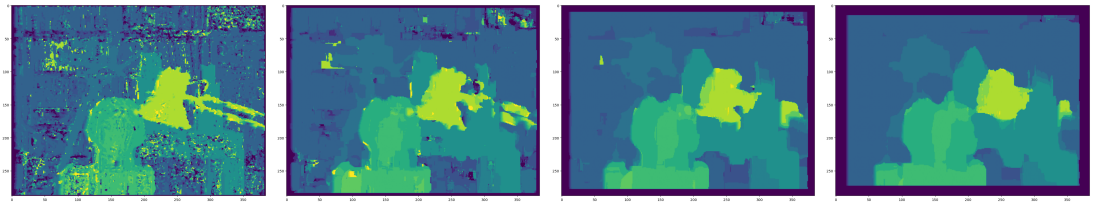

Fig. 8: Output using SSD and window sizes 3x3, 9x9, 21x21, 31x31

Using this method, we obtain results which preserve more detail and edges, but are way noisier than the ones we obtained using SAD and SSD with small window sizes (Figure 9).

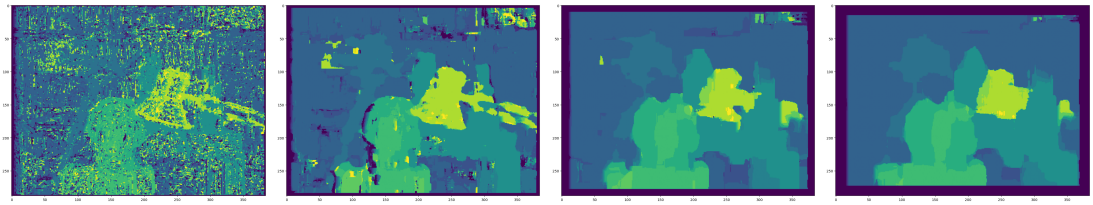

Fig. 9: Output using NCC and window sizes 3x3, 9x9, 21x21, 31x31

Apply to facade images On Table 1 we can see the Mean square error of each disparity map using the ground truth as a reference. As we can see, although NCC has a higher error in the 3x3 case, it decays rapidly using bigger window sizes. As seen qualitatively, SAD is the most robust out of all of them.

We can also see the effect of using different disparity values in Figures 10, 11 and 12. The effect can be clearly seen when using bigger window sizes: using higher disparities gives us a blurrier result, as we are taking into account pixel values that are further away.

Window size	**MSE SAD**	**MSE SSD**	**MSE NCC**
3x3	13.64	13.77	22.02
9x9	9.39	9.57	10.69
21x21	6.38	6.64	6.81
31x31	4.45	4.84	4.89

Table 1: MSE of each method

Adaptive support weights In this section we implement support weights based on the paper "Adaptative Support-Weight Approach for Correspondence search" [5]. Basically we focus on two

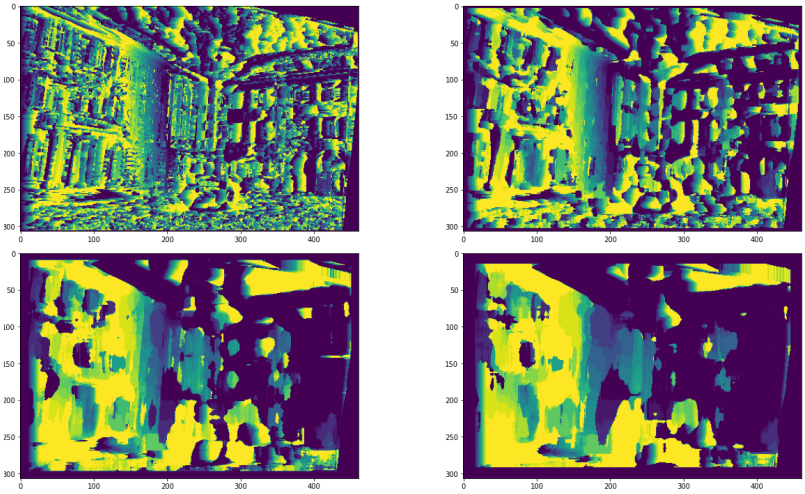

Fig. 10: Output using SSD and window sizes 3x3, 9x9, 21x21, 31x31, min_disparity = 0, max_disparity = 16

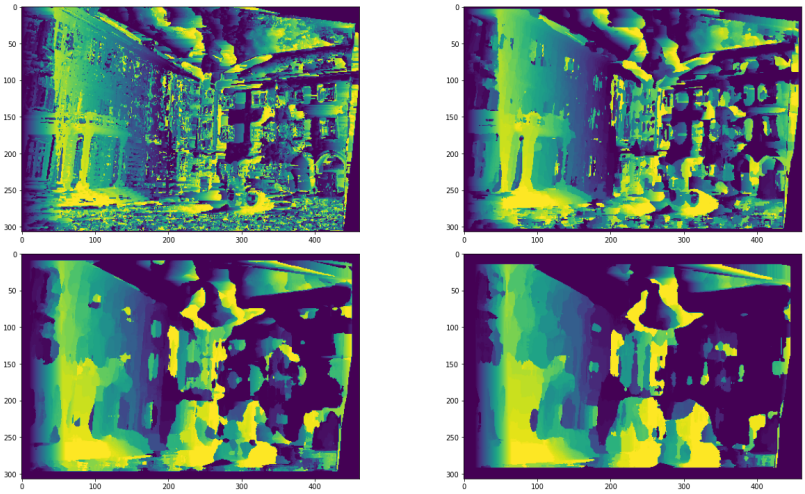

Fig. 11: Output using SSD and window sizes 3x3, 9x9, 21x21, 31x31, min_disparity = 0, max_disparity = 50

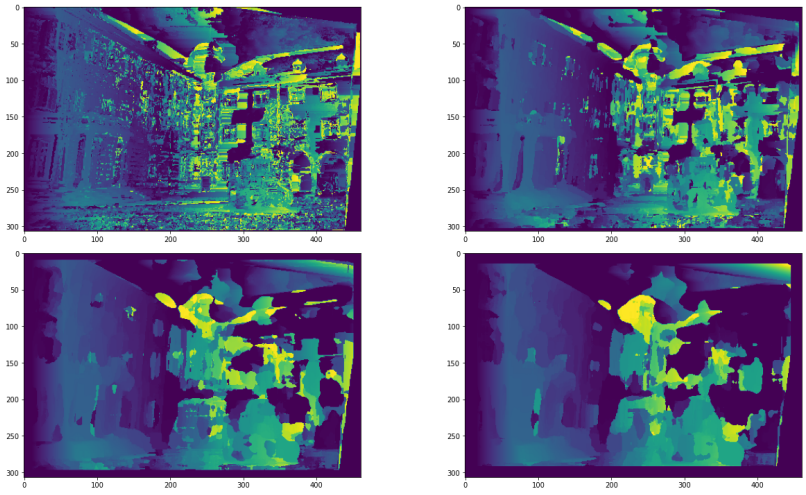

Fig. 12: Output using SSD and window sizes 3x3, 9x9, 21x21, 31x31, min_disparity = 0, max_disparity = 150

different ideas which are related to the gestalt principles of the human visual system, we will call those "exponential filters", because these ideas utilize the mathematical exponential function. The first one is about using support-weights based on gestalt grouping in the sense of color. We could interpret it as that pixel that are really near some n-dimensional colorspace are related somehow. This is the first exponential filter we use and shown in Equation (12) where the perceptual difference between two pixels can be expressed through the euclidean distance in the Lab colorspace or others such as RGB. In the original paper gamma is chosen as 14.

$$f_s = exp(-\frac{\triangle c_{pq}}{\gamma}) \tag{12}$$

The second idea is based on grouping by proximity. We could interpret this as that pixels that are closer to others in the spatial domain are more similar in some sense. Here, instead of using the color differences between pixels we use the differences in spatial domain. Similar to the case with color, we use the euclidean distance. Equation (13) shows this function.

$$f_p = exp(-\frac{\triangle g_{pq}}{\gamma}) \tag{13}$$

Following this, based on the two previous ideas we can create a new one, where the support weights depend on both the spatial and intensity (or color) differences of pixels. We could implement it with our previous exposed exponential filters as in Equation (14).

$$f_{sp} = exp(-\frac{\triangle c_{pq}}{\gamma_{col}}) * exp(-\frac{\triangle g_{pq}}{\gamma_{pos}}) \tag{14}$$

All these support weights are adaptive in the sense that depending on some circumstances (either spatial position of pixels or color for instance) they will modify their behavior. In our case the similarity between two patches or pixels will be weighted according to those terms. A visual

example of an adaptative weight based on color similarity can be seen in Figure 13. In it we can observe that in this case on the lower right pixels the blocks were more dissimilar in color, and this information will be used for the overall similarity metric between both blocks.

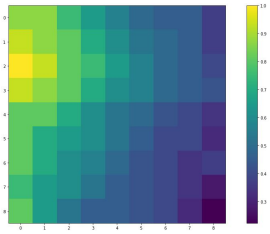

Fig. 13: An example adaptative weight between two different image blocks

Now, we could also implement the adaptative weights for black and white images. We could do this by just taking into account the similarity in proximity instead of the color, because there is not much more information in black and white than in the actual value of the pixel.

To demonstrate this, some results may be seen by computing the disparity maps on the images of Figure 14. This images are taken from [6]. We can observe in Figure 15 how the adaptative weights on color have taken into account the shadows of the image, and thus a different disparity map is computed from an image without those adaptative weights.

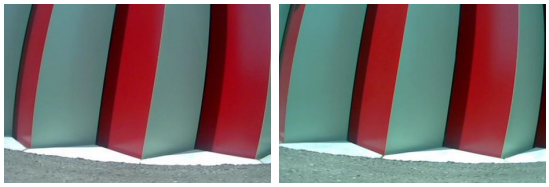

Fig. 14: Left and right images of repetitive structures and textures

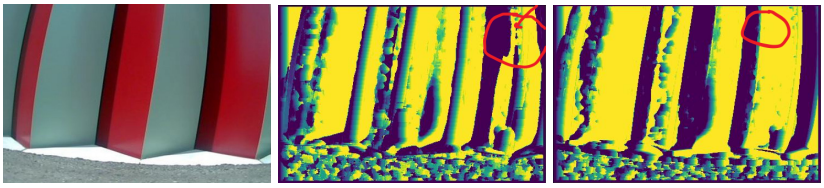

Fig. 15: Disparity maps of repetitive structures and textures. Depending on the adaptative weights different results are computed

3 Conclusions

In this work, we have implemented a triangulation technique, 3d reconstruction from two images and estimated depth maps with different methods. We have shown some applications for triangulation and some insight into the implementation and results of the algorithms are shown.

Some problems that appeared during the implementation of the exposed techniques are:

- We had some minor problems with code running slow. As exposed in previous classes, this could be solved by using vectorization.

Overall this has been an interesting project, where we have seen both the inner implementations for fundamental matrix estimation and their results. As a relevant bibliography, we have used the book Multiple View Geometry in Computer Vision by Richard Hartley and Andrew Zisserman [1].

References

1. Hartley, R., Zisserman, A. (2004). Multiple View Geometry in Computer Vision (2nd ed.). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511811685
2. T. Basha, Y. Moses, and S. Avidan. Photo Sequencing, International Journal of Computer Vision, 110(3), 2014.
3. B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In Proc. SIGGRAPH, 1996.
4. RoutedFusion: Learning Real-time Depth Map Fusion Silvan Weder, Johannes L. Schönberger, Marc Pollefeys, Martin R. Oswald May 2020.
5. Adaptive support-weight approach for correspondence search, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006.
6. Stereo Matching, https://campar.in.tum.de/twiki/pub/Chair/TeachingWs11Cv2/3D_CV2_WS_2011_Stereo.pdf. Last accesed 02 Feb