M6 3D RECOVERY OF URBAN SCENES - SESSION 4

AXEL BARROSO & SERGIO CASTRO & SERGIO SANCHO

GOAL

The goal of this session is to learn and compute the reconstruction over two images knowing the internal parameters.

1 INTRODUCTION

First of all, we need to set the triangulation between to different cameras where we know their projection matrices P and P' and 8 known correspondences between them.

Triangulation allows us to find the visual point X that is passed by the rays of the center of both cameras. In the real world, this point X is not the same for both cameras as they have noise, that is why we need to make the triangulation in order to find the point X that is closer to the real points x and x'.

In the triangulation method we are minimizing the expression of figure 1 applying the Homogeneous method (DLT algorithm).

$$min_{\boldsymbol{X}} \parallel A\boldsymbol{X} \parallel_2 with \parallel \boldsymbol{X} \parallel_2 = 1$$

Figure 1: Homogeneous Method DLT to be minimized

In this case, A is the system of equations formed by the correspondences and the projection matrices of both cameras. Its formula can be seen in figure 2.

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

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

Figure 2: System of equations

We build A from multiplying the different points x, x', P and P' with the Homography Matrix H, that can be found as showed in figure 3.

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

Figure 3: Homography Matrix

Where nx and ny are the size of the image (assuming that both images have the same size).

The main steps of the DLT algorithm are the following:

$$\text{where } \mathbf{A} = (\mathbf{A}'|\mathbf{a_4}), \text{ and } \mathbf{X} = (\tilde{\mathbf{X}}, 1)^T.$$

(i) Find the SVD $\mathbf{A}' = \mathbf{UDV}^T$.
(ii) Set $\mathbf{a_4}' = \mathbf{U}^T\mathbf{a_4}$.
(iii) Find the vector $\mathbf{Y}$ defined by $y_i = \frac{-a'_{4_i}}{d_i}$, $i = 1, \ldots, n$, where d_i is the i diagonal element of $\mathbf{D}$.
(iv) The solution is $\tilde{\mathbf{X}} = \mathbf{VY}$.

Figure 4: Steps of the DLT Algorithm

In order to check if the triangulation that we constructed work correctly, in figure 1 the results of the differences between the original points and the ones computed by triangulation are shown.

```
1.0e-14 *

-0.1998   -0.1221   -0.0500   -0.0555   -0.2665   -0.1332   -0.0444   -0.0111
-0.2331   -0.2220   -0.0056   -0.0444   -0.2276   -0.1221   -0.2248   -0.0444
-0.6217   -0.4441   -0.3997   -0.2220   -0.7550   -0.5329   -0.2665   -0.2665
```

Figure 5: Steps of the DLT Algorithm

As can be seen, the error is almost 0 in all of the different points computed, showing us that we can continue with the practicum as the triangulation method works correctly.

2 RECOVERY

After computing the triangulation function, we need to reconstruct the different points from the two different views of the cameras.

As explained in the previous section, we will apply triangulation knowing the 8 correspondences between the both cameras and its correspondence projection matrices.

As in the previous practicums, we will apply sift in order to compute and match the different keypoints of both images, applying RANSAC to avoid the outliers. This step can be seen in figure 6.

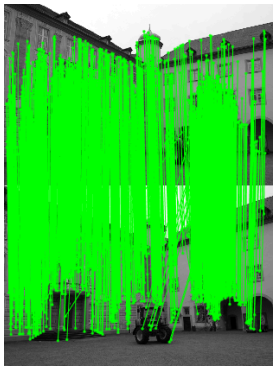


Figure 6: Keypoints correspondences

After that, we need the camera projection matrices. In order to compute them, we assumed that the first projection matrix is the origin of coordinates, and for the second one, we will need to compute the essential matrix and perform and SVD to E in order to find the 4 possible solutions. In figure 7 , the formulas for finding the essential matrix E and the 4 different possibilities of the SVD of E can be seen.

$$E = K'^{*}F^{*}K$$

$$P = K \cdot [I \mid 0]$$

$$P_1' = \left[UWV^T \mid +\mathbf{u}_3\right] \quad P_2' = \left[UWV^T \mid -\mathbf{u}_3\right]$$

$$P_3' = \left[UW^TV^T \mid +\mathbf{u}_3\right] \quad P_4' = \left[UW^TV^T \mid -\mathbf{u}_3\right]$$

Figure 7: E matrix, P projection matrix and P' 4 possibilities

In order to understand more the 4 different possibilities of the 2nd projection matrix, the figure 8 shows them.

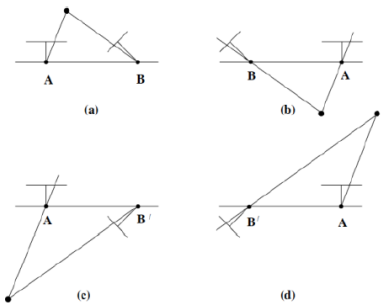


Figure 8: 4 different possibilities

In our case, in figure the 4 different possibilities are shown in blue and the P origin view is shown in red.

In order to see which one is the best correspondence for our origin, after selecting two matched Keypoints (one for each camera) and applying the

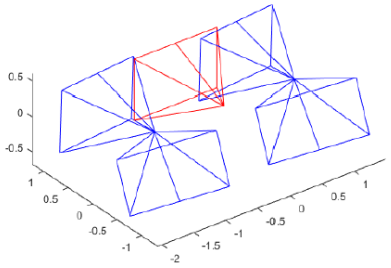


Figure 9: All the views

triangulation, we multiplied the Projection matrices of the both cameras with the Keypoinst and we check that the third component of each chosen Keypoints is not negative. In figure 12 the chosen view can be seen in green.

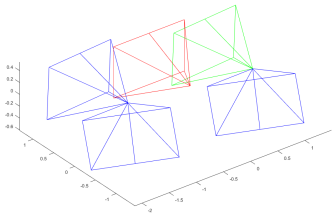
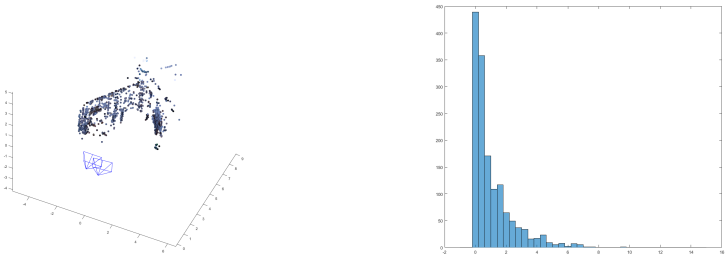


Figure 10: Chosen View

In figure 11 the 3D points computed can be seen and also the histogram of the reprojection error for each 3D point. The Reprojection error is computed as in figure .



(a) 3D Points **(b)** Histogram of Reprojection Error

Figure 11: Keypoints before and after RANSAC

$$Error = \sum_i d(x_i, \hat{x}_i)^2 + d(x'_i, \hat{x}'_i)^2$$

Figure 12: Reprojection Error Formula

3 DEPTH MAP COMPUTATION

In this part of the practicum, we will find the correspondence between points belonging to the same location that are represented in two different images. These points will have a disparity, which means the different position that they have in the both different views. Thanks to the disparity, we will be able to compute the depth of the different objects. We will find the correspondences by searching in horizontal lines in the both image views. The two views of the images can be seen in figure 13 .

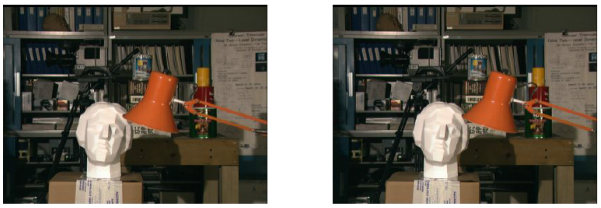


Fig. 5. Different views used on the disparity computation.

Figure 13: Views of the same image

3.1 Depth Map with SSD cost

IN this cost case, we will minimize it by minimizing the Sum of Squares Difference. Its formula can be seen in figure 14.

$$\sum_{(x,y)\in W} w(x,y)\,|I_1(x,y) - I_2(x_0+x, y_0+y)|^2$$

Figure 14: Sum of Square Differences

As expected, the results are windows dependent, where its size affects a lot to the different performances that we can achieve. With bigger windows we can blurr objects and erase their limits, but we don't get noise whereas with small windoes, we are more precise but the quantity of noise that we obtain is bigger.

In order to compare with the groundtruth we have compared them with the computations that has an error bigger than one.

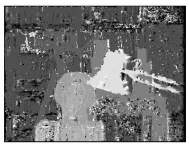


Figure 15: window: 3x3

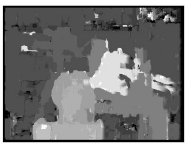


Figure 16: window: 9x9

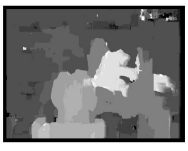


Figure 17: window: 15x15

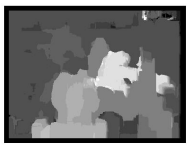


Figure 18: window: 20x20

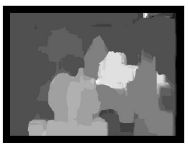


Figure 19: window: 30x30

Figure 20: window: 40x40

3.2 Depth Map with NCC cost

In this case we will be maximizing the correlation between instead of minimizing the error as seen before. The formula of the cost can be seen in figure 21.

$$\frac{\sum_{(x,y)\in W} w(x,y)\left(I_1(x,y)-\overline{I_1}\right)\left(I_2(x_0+x,y_0+y)-\overline{I_2}\right)}{\sigma_{I_1}\sigma_{I_2}}$$

Figure 21: NCC Cost

It is important to notice that after running the code for SDD and NCC we have notices that the computation time is highly longer with the NCC cost. Also, it has been very important to chose the correct windows size as can be seen in the next figures. Although, it is possible to see that the performance of the NCC cost is slightly lower than in the SDD cost.

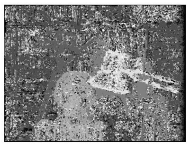


Figure 22: window: 3x3

Figure 23: window: 9x9

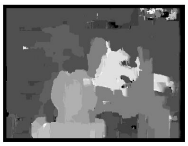


Figure 24: window: 15x15

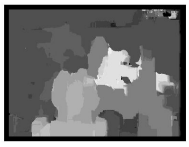


Figure 25: window: 20x20

Figure 26: window: 30x30

Figure 27: window: 40x40

3.3 Depth Map with bilateral weights

In this case, the method tries to minimize the cost by using weights based on color similarity and spatial distance. The cost formula can be seen in figure 28.

$$C(p,\hat{p}_d) = \frac{\sum_{q\in N_p,\hat{q}_d\in N_{\hat{p}_d}} w(p,q)w(\hat{p}_d,\hat{q}_d)c(q,\hat{q}_d)}{\sum_{q\in N_p,\hat{q}_d\in N_{\hat{p}_d}} w(p,q)w(\hat{p}_d,\hat{q}_d)}$$

Figure 28: Bilateral Cost Formula

Comparing this method, we have seen that the performances seems to see slightly similar to the NCC cost but worse than in the SDD one. The best result is obtained with a window of 30x30 and that the time spent to compute it has been also similar to the NCC cost. We have not been able to improve our results when it was supposed that this method was more robust and gave better performances than the ones used before. The results can be seen in the next figures.

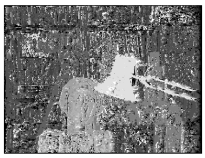


Figure 29: window: 3x3

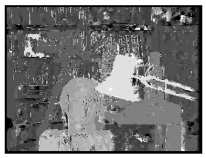


Figure 30: window: 9x9

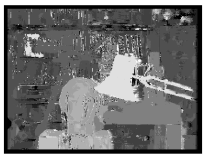


Figure 31: window: 15x15

Figure 32: window: 20x20

Figure 33: window: 30x30

Figure 34: window: 40x40

3.4 Depth Map with façade images

Finally, we will find the disparity maps in the Facade images with the three different methods tested in the previous sections. We have set different trials and at the end, the windows size that gave the best results are 20*20 with the NCC cost and the SSD cost and 40*40 with the Bilateral Weights.

In the figures 35, 36 and 37, we can see that both the SSD and NCC methods have delivered a good result, where they are making a good reconstruction of the left facade (in the rigth one the SSD works better), but in the bilateral weights seems that the noise that we get is big in both the left and rigth facade.

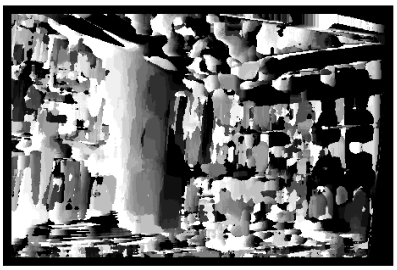


Figure 35: NCC cost in Facade with 20*20 Window

Figure 36: SSD cost in Facade with 20*20 Windows

Figure 37: Bilateral Weights in Facade with 40*40 Window

REFERENCES

[1] R.I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.