3D recovery of urban scenes: Reconstruction from two images

Josep Brugués i Pujolràs[1], Sergi García Sarroca[1], Òscar Lorente Corominas[1], and Ian Pau Riera Smolinska[1]

Universitat Autònoma de Barcelona, Bellaterra, 08193, Catalonia, Spain
{josep.brugues, sergi.garciasa, oscar.lorentec, ianpau.riera}@e-campus.uab.cat

1 Introduction

In this project, we triangulate the matching correspondences between two views of the same scene, by means of applying the Direct Linear Method (DLT). We will compute the camera matrices of these images given the Fundamental Matrix and the Intrinsic Matrix, and evaluate the triangulation method by estimating the reprojection error. Then we will apply the triangulation in order to create a 3D reconstruction from two views.

The second part of the project focuses on computing depth maps using local methods (SSD, NCC) and implementing bilateral weights on this mapping.

We also implemented two optional tasks: stereo computation with Belief Propagation and new view synthesis (morphing technique).

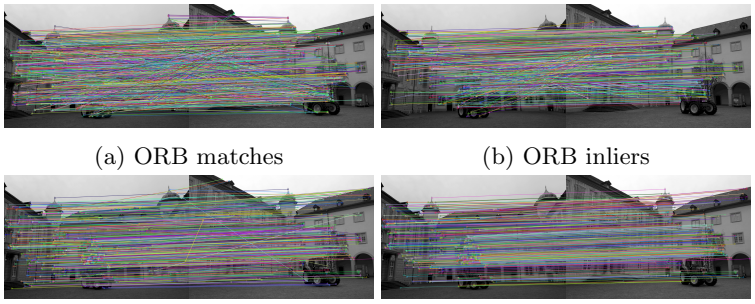
In this document, we present the methodology that was followed for each section, as well as the results obtained and the problems that have arisen.

2 Triangulation

Given two different matching points x and x', and knowing the camera matrices P and P', we aim to use the algebraic DLT method to triangulate the 3D points. To validate the implementation, we use a toy example where the euclidean points and cameras matrices are given. First, we project a 3D point with the camera matrices to obtain two 2D points $x = PX$ and $x' = P'X$. Then, we use these two 2D points to triangulate the corresponding 3D point $\hat{X}$, and we compare it to the original 3D point X. We obtain a zero error, which ensures that our method triangulates properly.

3 Reconstruction from two views

In this part, we work with the two gray-scale facade images. The first step is to obtain the image keypoints and matches, using both ORB and SIFT. Then, we estimate the fundamental matrix using the RANSAC robust method explained in previous weeks. The corresponding matches and inliers are presented in Fig. 1.



(a) ORB matches (b) ORB inliers

(c) SIFT matches (d) SIFT inliers

Fig. 1: Difference between ORB and SIFT keypoint detectors.

As observed, we obtain much more outliers using ORB keypoint detector, so we discard it. For this reason, we use SIFT to compute the matches and estimate the fundamental matrix F.

From the given camera calibration matrices, K and K' (in this case $K = K'$), and using the estimated fundamental matrix, we can compute the essential matrix $E = K'^T F K$. Applying the SVD decomposition to the essential matrix $E = UDV^T$, we can obtain four different pose estimations for the cameras as:

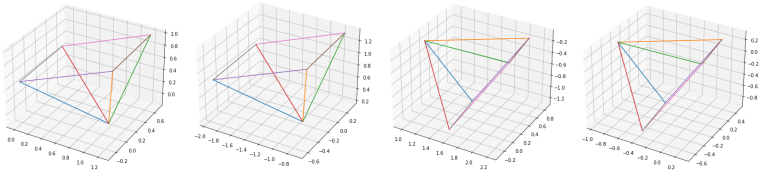
$$\begin{aligned} P'_1 &= \left[UWV^T \mid +\mathbf{u}_3\right] \quad P'_2 = \left[UWV^T \mid -\mathbf{u}_3\right] \\ P'_3 &= \left[UW^TV^T \mid +\mathbf{u}_3\right] \quad P'_4 = \left[UW^TV^T \mid -\mathbf{u}_3\right] \end{aligned} \tag{1}$$

where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

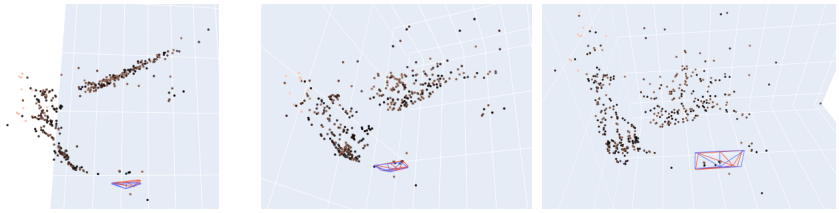
and u_3 is the last column of matrix U.

From the four possible matrices we can obtain, we use the one that makes geometrical sense in a realistic way, as the reconstructed point has to be in front of both cameras. That is, the z coordinate of the 2D point computed using the corresponding P' must be positive.



(a) $P2_1$ (b) $P2_2$ (c) $P2_3$ (d) $P2_4$

Fig. 2: Four different camera configurations obtained with each possible P'.



(a) (b) (c)

Fig. 3: 3D reconstruction of the two facade images from different view points.

We can see the four different cameras obtained with each P' on Fig. 2.

After deciding which is the correct P', we can use it together with P to triangulate 3D points from the two images using the corresponding 2D matches. We can see the resulting 3D reconstruction in Fig. 3, where the two cameras properly rotated and translated are also shown.

We can also analyze the 3D reconstruction quantitatively by computing the re-projection error. To compute it we use the following equation:

$$\sum_i d\left(\mathbf{x_i}, \hat{\mathbf{x}}_\mathbf{i}\right)^2 + d\left(\mathbf{x}'_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i}\right)^2 \tag{3}$$

where $\hat{x} = PX$ and $\hat{x}' = P'X$.

We present the histogram of re-projection errors for each 3D point in Fig. 4.

As observed, the mean error is quite high, as this method is sensible to outliers. Furthermore, due to the randomness of the RANSAC algorithm, the results change for each iteration, and so does the error.

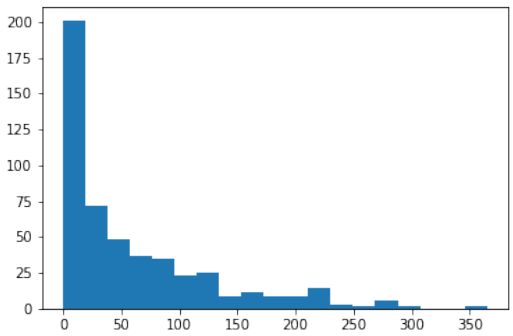


Fig. 4: Histogram of re-projection errors. Mean error: 60.9

4 Depth map computation

The goal of this exercise is to compute the disparity map from two rectified stereo images using Eq. 4.

$$disparity = x - x' = \frac{baseline * f}{z} \tag{4}$$

The basic idea of stereo matching is:

- Slide a window along the same line in the right image and compare its content to the reference window in the left image.
- The disparity is given by the pixel with the minimum matching cost.

There are multiple cost functions that can be used in stereo matching to compute the difference between the regions of interest of the two images. In this project, we use the sum of squared differences (SSD) and the normalized cross correlation (NCC), described by Eq. 5 and Eq. 6, respectively.

$$SSD(\mathbf{p}, \mathbf{q}) = \sum_{q \epsilon N_p} w(\mathbf{p}, \mathbf{q}) |I_1(\mathbf{q}) - I_2(\mathbf{q} + \mathbf{d})|^2 \tag{5}$$

$$NCC(\mathbf{p}, \mathbf{d}) = \frac{\sum_{\mathbf{q} \in N_p} w(\mathbf{p}, \mathbf{q}) \left(I_1(\mathbf{q}) - \bar{I}_1\right) \left(I_2(\mathbf{q} + \mathbf{d}) - \bar{I}_2\right)}{\sigma_{l1} \sigma_{l2}} \tag{6}$$

where

$$\bar{I}_1 = \sum_{\mathbf{q} \in N_\mathrm{p}} w(\mathbf{p}, \mathbf{q}) I_1(\mathbf{q}) \qquad \bar{I}_2 = \sum_{\mathbf{q} \in N_\mathrm{p}} w(\mathbf{p}, \mathbf{q}) I_2(\mathbf{q} + \mathbf{d})$$

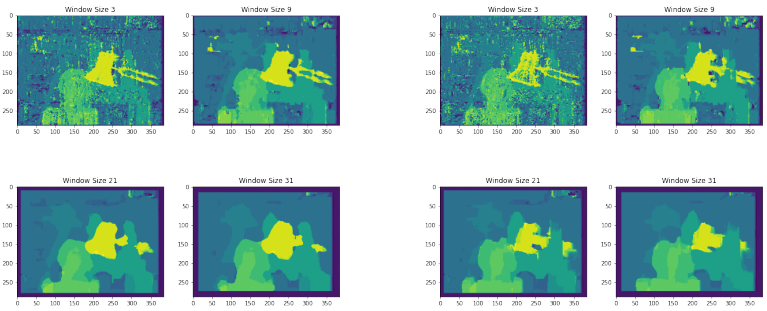
$$\sigma_{l_1} = \sqrt{\sum_{\mathbf{q} \in N_\mathrm{p}} w(\mathbf{p}, \mathbf{q}) \left(I_1(\mathbf{q}) - \bar{I}_1\right)^2} \qquad \sigma_{l2} = \sqrt{\sum_{\mathbf{q} \in N_\mathrm{p}} w(\mathbf{p}, \mathbf{q}) \left(I_2(\mathbf{q} + \mathbf{d}) - \bar{I}_2\right)^2}$$

The results obtained with both approaches and different window sizes are presented in Fig. 5.

We can observe that with smaller window sizes we obtain more details, but the results are noisier. As we increase the window size we lose detail and obtain smoother disparity maps. Furthermore, if the window size is too large, the objects end up merging. On the other hand, we can observe that SSD provides better qualitative results, as the objects regions are more homogeneous.

SSD & NCC with bilateral weights

In the previous approach, each pixel on the windows is given the same weight. The idea behind bilateral weights is to assign to each neighboring pixel a different importance by taking into account their color similarity and geometric proximity with respect to center of the window. The weight computation for gray scale images is described by Eq. 7.



(a) (b)

Fig. 5: Scene1 - Disparity results with (a) SSD and (b) NCC

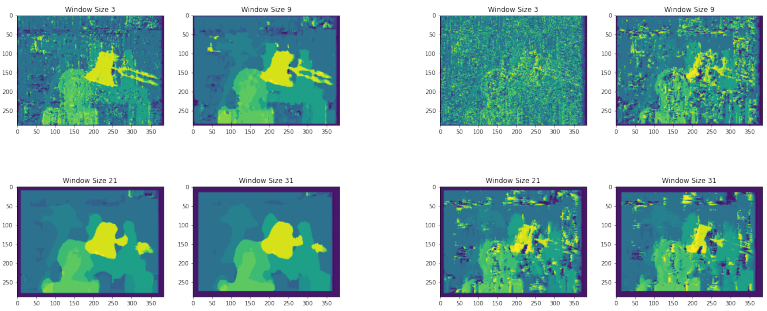
$$w(\mathbf{p},\mathbf{q}) = w_{col}(\mathbf{p},\mathbf{q})w_{pos}(p,q) = exp\left(-\frac{\Delta c(\mathbf{p},\mathbf{q})}{\gamma_{col}}\right) exp\left(-\frac{\Delta g(\mathbf{p},\mathbf{q})}{\gamma_{pos}}\right) \quad (7)$$

where

$$\Delta c(\mathbf{p},\mathbf{q}) = ||I(\mathbf{p}) - I(\mathbf{q})\Delta g(\mathbf{p},\mathbf{q}) = ||\mathbf{p}-\mathbf{q}||_2 = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2}$$

The results are presented in Fig. 6. We can observe that using bilateral weights with the SSD algorithm slightly improves the results, as the depth of each object is more accurately represented. However, using bilateral weights does not improve the NCC method, and we end up with noisy estimations of the depth. We believe that the results obtained with bilateral weights are not as good as expected due to the fact that bilateral weights give more importance to pixels similar in color. Since we are using gray-scale images, we lose two dimensions of information and the method fails to estimate the optimal weights. For example, it is easier to correctly perceive depth between a red and a blue object than between two gray ones.

We also present a quantitative comparison between each method in Tab. 1, which validates our visual observations. SSD performs better than NCC, and the results are improved as the window size is increased (even if that comes with a drawback, which is that we lose details of the image). Moreover, bilateral weights slightly improves when using SSD with larger windows, although it significantly deteriorates the results with NCC.



(a) (b)

Fig. 6: Disparity results with (a) SSD and (b) NCC, both with bilateral weights

Table 1: Prediction error

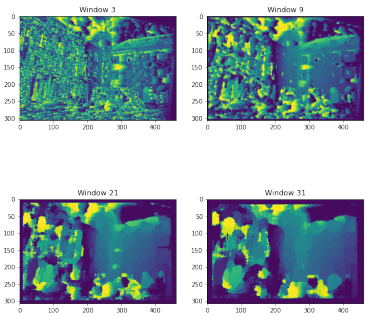
Cost	Windows size	Bilateral Weights	
		Yes	No
SSD	3	15.09	14.9
	9	9.55	9.54
	21	6.08	6.12
	31	3.88	4.01
NCC	3	29.39	17.77
	9	22.47	10.22
	21	14.46	6.48
	31	10.81	4.48

Depth map computation with facade images

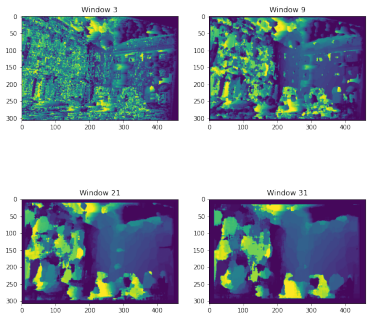
We have also calculated the disparity map for the facade images, which are presented in Fig. 7. In this case, we tried different window sizes, maximum disparity values and the two cost functions.

As aforementioned, smaller window sizes provide noisy results with much more details, and we can differentiate some specific objects in the scene, such as windows or walls. On the other hand, the noise is decreased when using larger window sizes, but we lose detail as well. Changing the maximum disparity does not affect much, as the disparity between objects that are far away is small.

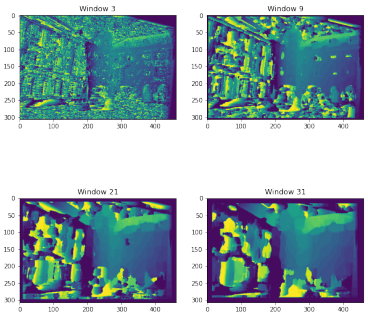
In this case, we obtain much worse results compared to the ones from scene1. This is due to the fact that the objects from which we try to infer depth are further away, so the changes in disparity are smaller and thus harder to detect. Moreover, we have repetitive patterns (windows and the texture of the building), so the method fails to find the disparity minimum correctly.



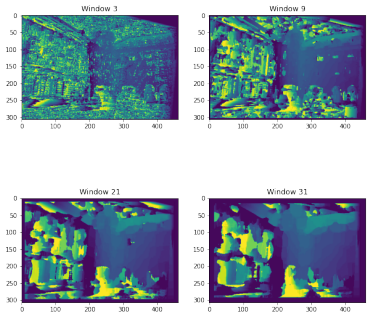
(a) SSD - Max disparity of 32



(b) SSD - Max disparity of 48



(c) NCC - Max disparity of 32



(d) NCC - Max disparity of 48

Fig. 7: Facade images - Disparity maps estimation

5 Optional Tasks

5.1 Stereo computation with Belief Propagation

In this method we treat the image as a graph and approach the problem of recovering an accurate disparity map as an energy minimization problem:

$$\mathcal{E}(\mathcal{L}) = \sum_{p \in \mathcal{P}} \mathcal{D}_p\left(l_p\right) + \lambda \cdot \sum_{p,q \in \mathcal{N}} \mathcal{V}\left(l_p - l_q\right) \tag{8}$$

where $lp_{\in 0,1,2,\ldots,dmax-1}$ is the disparity value of pixel p, P is the set of pixels in the image, N is the set of undirected edges in the four-connected image grid

graph, $Dp(l_p)$ is the cost of assigning label-disparity l_p to pixel p, and $V(l_p - l_q)$ is the cost of assigning labels l_p and l_q to two neighbor pixels. The data cost $Dp(l_p)$ is defined as the truncated absolute intensity difference:

$$\mathcal{D}_p\left(l_p\right) = \min\left(\frac{1}{3}\left\|I_{left}(y,x) - I_{right}\left(y, x - l_p\right)\right\|_1, \tau\right) \tag{9}$$

where $I(y,x) \in \mathbb{R}^3$ is the RGB color of the pixel and $\|\|$ is the $l_1 norm$. As measure of difference between labels $V(l_p - l_q)$ we use the Potts model:

$$\mathcal{V}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

Finding the disparity map that minimizes the above energy, corresponds to Maximum A Posteriori (MAP) estimation problem for an appropriately defined Markov Random Field (MRF). To approximate the MAP solution, We use the min-sum Loopy Belief Propagation (LBP) algorithm.

The results are presented in Fig. 8. We can observe that we obtain better results than using the previous methods, specially in the scene1 image. As aforementioned, the results are not that good for the facade image. Additionally, we plot the evolution of the cost function for every iteration, shown in Fig. 8c and Fig. 8d respectively. We can observe that for the first case, the cost function converges in less iterations, and the minimum error is also smaller, which correspond with the visual results.

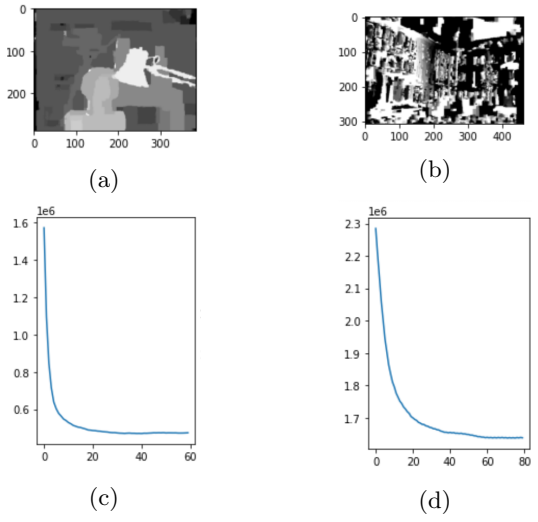


Fig. 8: Stereo computed images: (a) scene 1 and (b) facades, and cost function evolution: (c) scene 1 and (d) facades.

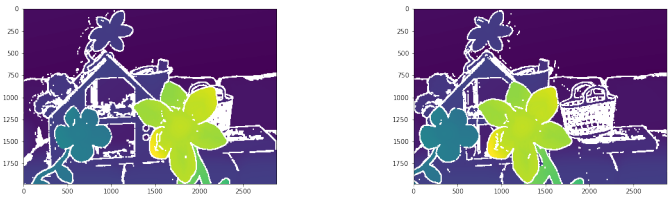


Fig. 9: From left to right: disparity map of the left and right image.

5.2 New view synthesis

In this section, we implement the new view synthesis method presented in [1] to create new views from a pair of rectified stereo images without doing any prewarp nor postwarp. We are facing the special case of two parallel views, in which the camera moves parallel to the image plane. In this case, and using the ground truth disparities of the left and right images, d_l and d_r respectively (Fig. 9), we use:

$$\mathbf{p} = (1-s)(x,y) + s\,(x - d_\ell(x,y), y) \tag{11}$$

where s is the interpolation factor which defines the position of the new view with respect to the other two views, to compute:

$$I(\mathbf{p}) = (1-s)I_\ell(x,y) + sI_r\,(x - d_\ell(x,y), y) \tag{12}$$

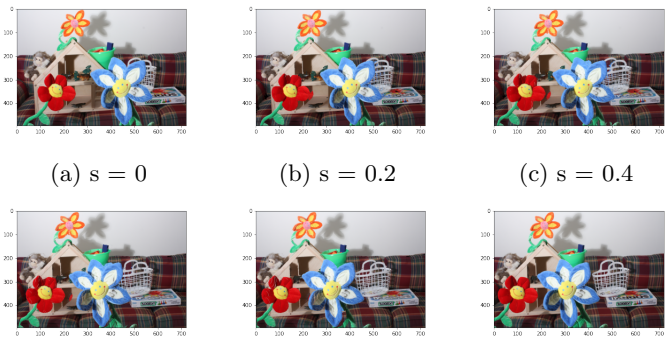
We must take into account occlusions using:

$$\|d_\ell(x,y) - d_r\,(x - d_\ell(x,y), y)\| \leq \epsilon \tag{13}$$

where ϵ is a tolerance factor manually defined. If there is an occlusion, we directly assign to the pixel of the new view image the value of the pixel in the image in which it is not occluded.

We generated nine new views using different values of $s = 0.1, 0.2, ..., 0.9$, where the new image is closer to the left (right) image for small (large) values of s. We attach an animated *GIF* in the notebook and in the zip folder submitted to the task. A sample of our results is shown in Fig.10

As observed, the algorithm described in [1] has been correctly implemented, as we manage to generate new views using only two images.



(a) s = 0 (b) s = 0.2 (c) s = 0.4

(d) s = 0.6 (e) s = 0.8 (f) s = 1

Fig. 10: New views generated using different interpolation factors

6 Conclusions

In this project we have worked on the triangulation of 3D world points from a pair of images and some of its applications, such as the 3D reconstruction.

The triangulation method is an efficient way to estimate the euclidean coordinates of a point, as it has been demonstrated using a toy example.

To use it in a practical example we needed to estimate the camera projection matrices. To do so, we used the robust 8-point algorithm presented in the previous week to estimate the fundamental matrix F. As in previous weeks, we compared the results obtained using ORB and SIFT keypoint detectors. We used the latter as we obtained a large number of outliers using ORB, so the estimation of F was not accurate.

Using triangulation, we were able to create a cloud of points estimating their position in the 3D world. Although we could imagine the relation between the obtained visualization and the original image, the results were a bit poor. To have a proper reconstruction, a significant amount of inliers is needed, and SIFT was failing to do that. Another problem was the randomness of RANSAC, which would make the results obtained inconsistent with every execution.

Furthermore, we implemented depth map computation by using local methods and two cost functions: SSD and NCC. When computing the disparity maps, we were able to observe the difference of performance between the 2 cost functions and different window sizes. From this results we can conclude that the smaller the window, the more detail in the depth map, but also the more noise. On the other hand, the bigger the window, the smoother the disparity at the expense of losing detail. Moreover, we can observe that the edges of the objects in the disparity could end up merged.

In addition, we implemented the depth estimation using the Loopy Belief Propagation algorithm, which treats the image as a graph and approach the

problem recovering an accurate disparity map as an energy minimization problem. The results show that this algorithm outperforms the previous methods.

Finally, we implemented the view synthesis by applying the interpolation method explained in [1]. The results are quite impressive, as we managed to generate new views using only a pair of rectified stereo images taken from a different angle. In this case, we created an animated *GIF* to have a visual representation of the results. We also observe artifacts in some parts of the image, as this method is far from perfect, but it is good enough for our specific problem.

References

1. S. M. Seitz and C. R. Dyer, "View morphing," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 21–30. [Online]. Available: https://doi.org/10.1145/237170.237196