3D recovery of urban scenes: Homography estimation and applications

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, the homography relating two images is estimated from the keypoint correspondences between them. After obtaining the homography, different applications are tested: image mosaic, camera calibration, logo detection and logo replacement. 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. The core algorithm to complete the work in this week's project is the normalized DLT algorithm to estimate the homography.

2 Homography estimation with the DLT algorithm

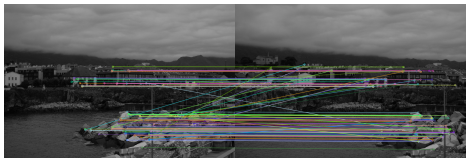
In this exercise we estimate the homography that relates two images using RANSAC and the normalized DLT algorithm.

2.1 Compute image correspondences

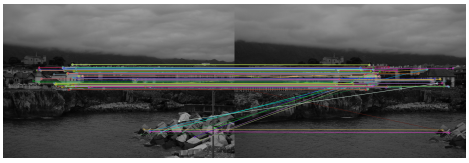
In order to compute the homography H that relates two images, we first need to compute 2D point correspondences between those images. In order to do so, we compute keypoints in both images using the ORB keypoint detector. In situations where ORB didn't detect keypoints accurately, we tried using other detectors such as SIFT and SURF. Then, we match those keypoints using K-NN. Each match gives a point correspondence. Fig. 1 shows an example of keypoint matching between two different images from the same scene but taken from different viewpoints. Notice that there are some mismatches, so we could adjust the distance threshold to improve the keypoint detection or matching. However, most of the keypoints are matched correctly, so we can properly estimate H using RANSAC.

2.2 Estimate the homography between image pairs

Once the 2D point correspondences are computed, we can use RANSAC combined with the normalized DLT algorithm to estimate the homography H. For

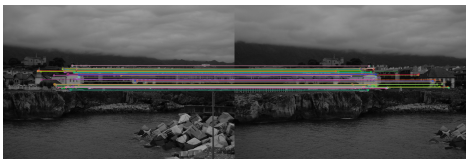


(a) Orb matching between images 1 and 2



(b) Orb matching between images 2 and 3

(c) DLT matches between images 1 and 2



(d) DLT matches between images 2 and 3

Fig. 1: Image correspondences

each iteration, RANSAC selects a random sample of 4 correspondences and the algorithm computes the homography using the normalized DLT algorithm. For each homography, the number of inliers is also computed, and the homography with the largest number of inliers is returned. Fig. 1c shows the matches that are considered inliers and thus were used to compute H. We can observe how the miss-matched keypoints of the rocks on 1a (diagonal lines to the houses) disappear in 1c after the DLT, as they are not considered inliers.

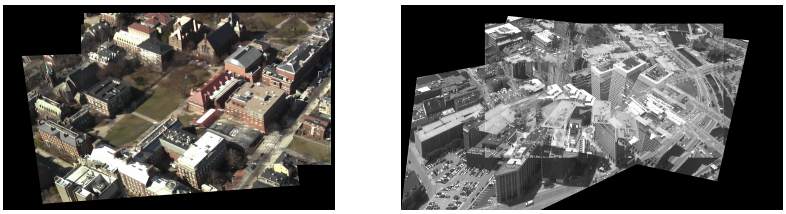
3 Application: Image mosaics

One of the applications of finding homographies between images is to build image mosaics. Once the homographies between one or more pairs of images are computed, they can then be used to transform images and fuse them into a new image, thus forming a mosaic. In this exercise we compute homographies to build panoramas from 4 different image sets. In order to properly create the mosaic, the relative position between the images in the real world needs to be taken into account. In our case we are using 3 images, so we need to know which image is in the left hand side, which one is on the right hand side and which one is at the center. The situation of the images conditions the H to use to transform the images: the left image uses the estimated H, the right image uses the inverse of the estimated H (because the homography was computed from the center image to the right one), while the central image is left unmodified (the homography is an identity matrix).

Fig. 2a presents the mosaic created from the images from the previous exercise. As it can be seen, the results are quite good: the high number of correct point correspondences results in correctly estimated homographies, so each image is transformed accordingly to form the resulting panorama.

Fig. 2b shows the mosaic created from the castle image set. It can be observed that while the building is well transformed, the tractor almost disappears. This

(a) Llanes (b) Castle



(c) Aerial Site 13 (d) Aerial Site 22

Fig. 2: Image mosaics

is due to the fact that our function gets the maximum between the 2 overlapped transformed images, and being the wall white, the tractor's colors are discarded.

Fig. 2c presents the mosaic created from images on the Aerial Set 13. It can be seen that the panorama is build correctly, as the image pairs are similar enough to obtain a high number of correct point correspondences, and thus the computed homographies are well estimated.

Finally, Fig. 2d shows the mosaic created from images on the Aerial Set 22. The resulting panorama is incorrect because the homographies are not estimated accurately. We can also observe this in Fig. 3b, where the inliers are not correct correspondences. Using another keypoint detector (in the notebook we used SIFT), the correspondences were correct, but the image mosaic was still wrong. The reason is that we cannot relate the images in the Aerial Set 22 with an homography, because the images haven't been taken by rotating a camera about its centre and they don't project the same plane in the 3D scene. For example, two different images might show opposite facades of the same building.

(a) (b)

Fig. 3: Aerial Set 22 - Inliers between (a) center and right and (b) left and center images.

Table 1: Geometric error before and after the refinement

H	Geometric error	
	Original	Refined
H_{12}	8963534	16
H_{23}	43022747	27

4 Refinement of the estimated homography with the Gold Standard algorithm

The previous homographies can be refined by minimizing the reprojection error, which is known as Gold Standard algorithm. This problem can be represented as a non-constrained minimization problem in Eq. 1.

$$\min_{\hat{H},\hat{x}_i} \sum_i d\left([x_i], [\hat{x}_i]\right)^2 + d\left([x_i'], \left[\hat{H}\hat{x}_i\right]\right)^2 \tag{1}$$

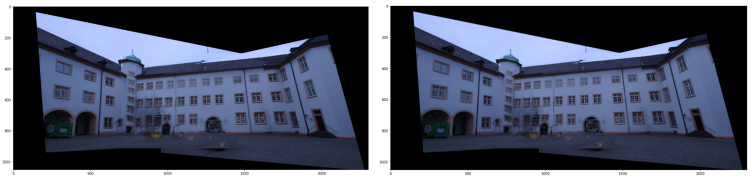
To do so, we use the function *least_squares* to minimize a geometric cost function based on Eq. 1 using Levenberg-Marquardt algorithm as the optimization method. The variables that are passed to the *least_squares* function and the geometric cost function are explained deeply in the notebook file.

We refine the homographies estimated in the castle image set to see if the resulting mosaic is improved. In Tab. 1 we present the reprojection (geometric) error before and after the refinement, where we can see that the error is correctly minimized.

The keypoint locations before and after the refinement are also shown in Fig. 4, where we can see that there is not a big difference between them. On the other hand, the resulting mosaic hasn't improved at all (Fig. 5). The reason for this is that the minimization is performed using the points that are correspondences (and inliers of the RANSAC estimation of H), so the refinement is limited to the initial estimation of the homography.

(a) Image 0014_s (b) Image 0015_s

Fig. 4: Keypoint locations before (blue) and after (purple) refinement - Castle



(a) (b)

Fig. 5: Castle mosaic (a) before and (b) after refinement

5 Optional. Application: Calibration with a planar pattern and augmented reality

5.1 Camera calibration

Homographies can be used for camera calibration with a planar pattern using Zhang's algorithm [1].

We can model a camera as the relationship between a 3D point X and its image projection x, as $x = PX$. We can decompose $P = K[R|t]$, where K is the matrix containing the internal parameters of the camera, and R and t contain the external parameters of the camera pose (position and orientation of the camera in the reference system of the world).

First of all we add some constraints. We consider that the world reference is attached to the pattern so that the pattern plane is $z = 0$. This allows us to rewrite $H = K(r1 \quad r2 \quad t)$. In addition, R is an orthonormal matrix, so $r1 * r2 = 0$ and $r1 * r1 = r2 * r2 = 1$. Therefore, we get the following equations on K:

$$\begin{gathered} h_1^T K^{-T} K^{-1} h_2 = 0 \\ h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \end{gathered} \tag{2}$$

To find K, we use the image ω of the absolute conic, as $\omega = K^{-T}K^{-1}$. We can rewrite the equations in 2 as:

$$h_i^T \omega h_j = \begin{pmatrix} h_{1i} & h_{2i} & h_{3i} \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ x_2 & x_4 & x_5 \\ x_3 & x_5 & x_6 \end{pmatrix} \begin{pmatrix} h_{1j} \\ h_{2j} \\ h_{3j} \end{pmatrix} = \mathbf{v}_{ij}^T \mathbf{x} \tag{3}$$

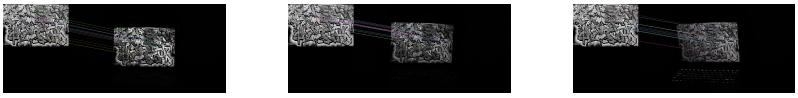
with

$$\begin{aligned} \mathbf{v}_{ij}^T = (&h_{1i}h_{1j}, h_{1i}h_{2j} + h_{2i}h_{1j}, h_{1i}h_{3j} + h_{3i}h_{1j}, \\ &h_{2i}h_{2j}, h_{2i}h_{3j} + h_{3i}h_{2j}, h_{3i}h_{3j}) \end{aligned} \tag{4}$$

Therefore, for every image we have the following two equations:

$$\begin{gathered} \mathbf{v}_{12}^T \mathbf{x} = 0 \\ \left(\mathbf{v}_{11}^T - \mathbf{v}_{22}^T\right) \mathbf{x} = 0 \end{gathered} \tag{5}$$

Since there are six unknowns, we need at least three views of the planar pattern. We used the images shown in Fig. 6 to find the homographies that relate them to the template.



(a) View 1 (b) View 2 (c) View 3

Fig. 6: Matches between template image and the three views of the planar pattern

With the three pairs of Eq. 5, we create the matrix V. Then we use the SVD decomposition to obtain the last column of the matrix $\hat{U}$ (being $V = UD\hat{U}^T$) as our solution vector. Finally, we can obtain the image of the absolute conic, ω, using the values on the vector.

From ω, we can use the Cholesky decomposition to retrieve the upper-triangular matrix of internal parameters, K:

$$K = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where f_x and f_y are the focal lengths on the x and y axis, respectively, p_x and p_y are the coordinates of the principal point and s is the skew factor. In our case, we have a focal length $f_x = 3.35\text{x}10^3$ and $f_y = 3.38\text{x}10^3$. Notice that both values differ due to errors in the calibration process and distortion. The skew coefficient $s = 6.25$ is also related to this radial distortion, which results in non-squared pixels. This might happen as a result of taking an image of an image, where the axis of the lens is not parallel to the image plane, as in our case. Finally, the principal points in our case are $p_x = 1\text{x}10^3$ and $p_y = 4.73\text{x}10^2$.

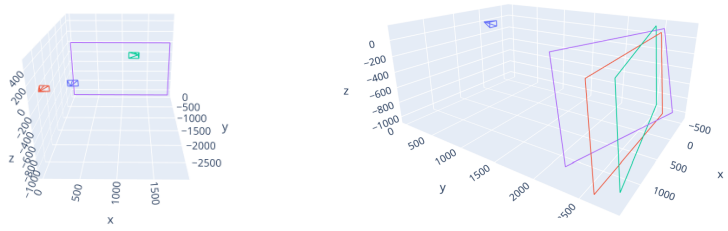
Notice that ω must be positive definite in order to decompose it using Cholesky factorization. In some cases, due to the randomness and errors introduced by our algorithm to estimate the homographies, ω might not be positive definite, so we have to iterate the process until this problem is solved.

Once we have the internal parameters we can retrieve the external parameters as:

$$\begin{gathered} \mathbf{r}_1 = \frac{K^{-1}\mathbf{h}_1}{\|K^{-1}\mathbf{h}_1\|}, \quad \mathbf{r}_2 = \frac{K^{-1}\mathbf{h}_2}{\|K^{-1}\mathbf{h}_2\|}, \quad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} = \frac{K^{-1}\mathbf{h}_3}{\|K^{-1}\mathbf{h}_1\|} \end{gathered} \tag{7}$$

Now that we have the external parameters we can compute the optical center and approximate the position of the camera in the real world. We obtain the optical center by computing the SVD decomposition of P, $P = UDV^T$. The optical center homogeneous coordinates are obtained as the singular vector associated to the smallest singular value (last column of V), converted then to Cartesian coordinates for its graphical representation.

We can either represent the position of the cameras with respect to the plane, as seen in Fig: 7a, or consider the camera fixed and the image plane moving, as in Fig: 7b.



(a) Cameras positions (b) Planes positions

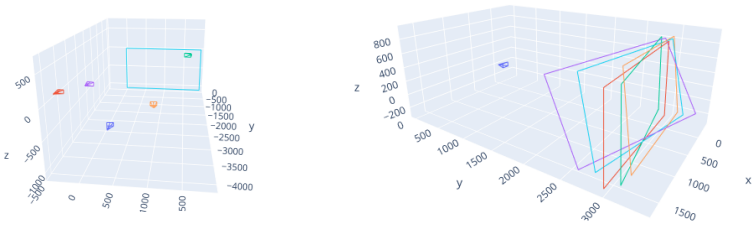
Fig. 7: Pose (3 cameras)

We can also use more than 3 images to calibrate, although we need to tweak the parameters a little to find good point correspondences. The results obtained with 5 pairs of equations are shown in Fig. 8. Using more views of the planar pattern, the errors in the calibration and the distortion observed in the intrinsic parameters should be reduced.

5.2 Augmented reality

Once the camera is calibrated and we have recovered the relative pose between the camera and the planar pattern, we can properly insert a virtual object maintaining the perspective sense with rest of the scene, as seen in Fig. 9.

To insert the virtual object in the image, we first define the corners of the cube and then project them onto the image plane using the projection matrix estimated for each of the image views. Notice that a 3D cube has eight corners,



(a) Cameras positions (b) Planes positions

Fig. 8: Pose (5 cameras)

(a) (b) (c)

Fig. 9: Virtual object on image view (a) 1, (b) 2 and (c) 3

but here we use 20 corners of five planes of the cube to later draw the projected 20 lines and thus represent the inserted object properly. We add an offset (related to the size of the template image) to these corners so that the cube is centered at the planar pattern.

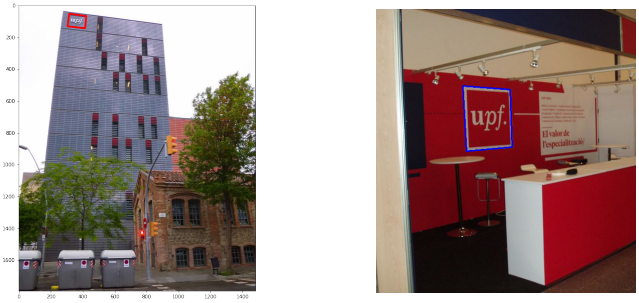
In order to get an effective augmentation of the real world, the real and virtual objects must be accurately positioned relative to each other. Because of that the pose and optical properties of the real and virtual camera must be the same. Therefore, we need the intrinsic camera parameters to project 3D points onto the image plane.

6 Optional. Application: Logo detection

The goal of this exercise is to detect the UPF logo in two images, one of a building and another one of a stand, both manually by selecting the 4 corners of the logo and automatically using keypoint detectors.

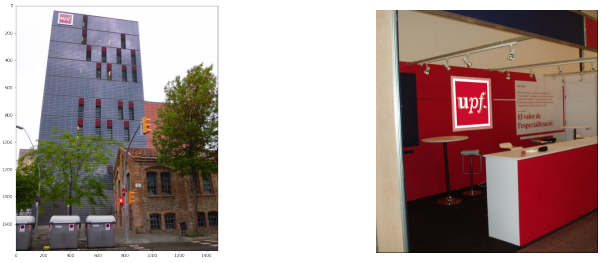
6.1 Manual detection

We manually select the corners of the UPF logo in the target image. Then, as we know the size of the template image *logoUPF*, we also have the corners of the



(a) (b)

Fig. 10: Manual logo detection in (a) building and (b) stand images



(a) (b)

Fig. 11: Manual UPF logo insertion in (a) building and (b) stand images

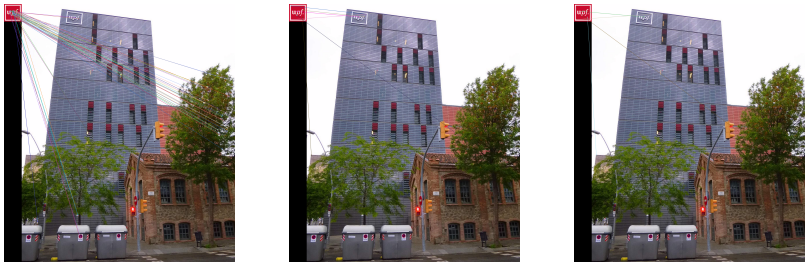
logo in this image, so we have four correspondences between both images. We can use these point correspondences to estimate the homography that relates them with the normalized DLT algorithm.

Using the estimated homography we can *detect* the logo by projecting the corners of the *logoUPF* image onto the target image and finally draw a polygon that highlights the detection (Fig. 10). The homography can also be used to create a mosaic and accurately insert the whole *logoUPF* image into the target image, as presented in Fig. 11.

Here, we manually selected the exact pixels that correspond to the corners of the UPF logo in the target image and used this correspondences to estimate H. In a different scenario, maybe we only have approximate guesses of these corners, so the correspondences may not be accurate enough to precisely estimate H. One way to solve this would be to compute keypoints in both images but restricting the region of the target image to the inside of these guesses of the corners. Then, using these keypoints, we could estimate H using RANSAC and DLT. This additional case is presented in the notebook.

(a) Initial good matches (b) Inliers (c) Drawing the detection

Fig. 12: Automatic logo detection in the stand image



(a) (b) (c)

Fig. 13: Building image: (a) matches with ORB, (b) matches with SIFT and (c) inliers with SIFT

6.2 Automatic detection

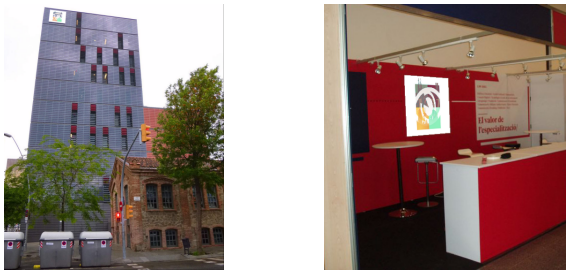
We can detect a logo in a target image automatically by using a keypoint detector. First, we detect and compute the keypoints in the *logoUPF* and target images, and then we use K-NN to find matches. If the matches are good enough, we can use RANSAC combined with the normalized DLT algorithm to estimate the homography that relates both images. This is process is presented for the stand image in Fig. 12.

The logo is automatically detected in the stand image, but in this case the result is not as accurate as the manual detection one, because here we're estimating H using correspondences that might be wrong or inaccurate.

In the case of the building image, the logo cannot be automatically detected because the matching process fails (there are too many textures in the building image that are wrongly matched with the template logo), as presented in Fig. 13a. We also tried with SIFT descriptor, and even if the initial matches are better (Fig. 13b), the estimation of H is not performed correctly and the inliers are not good either (Fig. 13c).

7 Optional. Application: Logo replacement

The goal of this exercise is to replace the UPF logo that has been detected in the previous images with the MCV logo. Once the homographies are estimated, we can use them to project the MCV logo onto the target images, so that the new logo is translated to its new position and transformed accordingly to fit the area of the old logo. Notice that the MCV logo image must be resized to the size of the UPF logo image. In Fig. 14 we use the homographies estimated with the manual detection, as they are more accurate and the results are satisfactory. In the case of automatic detection, it works for the stand image but not for the building one, as the homography is not correctly estimated.



(a) (b)

Fig. 14: MCV logo replacement on (a) building and (b) stand images

8 Conclusions

In this week's lab we have worked on homography estimation and some of its applications, such as constructing image mosaics, calibration with planar patterns and finding and replacing logos on an image. The results obtained have mainly been certified qualitatively, and on some cases such as the refinement with the Gold Standard Algorithm, quantitatively.

Despite successfully implementing all the algorithms, we have had some trouble obtaining good results in some cases. When constructing a mosaic with images from the aerial site 22, even if we have good point correspondences with SIFT detector, the images are not from the same 3D plane, so we cannot relate the images with an homography. We also had problems detecting automatically a logo on the UPF building due to the difficulty on finding good 2D point correspondences between the images. Finally, it is important to note that we were able to reduce the error on the homography estimation using the Gold Standard algorithm, so the quantitative results were good. Despite that, the qualitative results appear the same as using the normalized DLT algorithm.

As a take home message, we learned that homography estimation is very important in the field of computer vision, and good correspondences are essential in all the studied applications such as camera calibration, among others.

References

1. Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.