M4 - Lab 2: Homography estimation and applications

Sergio Montoya, Laia Albors, Ibrar Malik, Adam Szummer

Pompeu Fabra University
{ sergio.montoyadepaco, laia.albors, ibrar.malik, adam.szummer
}@e-campus.uab.cat

1 Introduction

This week's goal is to estimate the homography that relates two images using the robust DLT algorithm described in section 2. The estimated homography can be further refined with a non-linear optimization method consisting of minimizing a geometric distance, explained in section 4. Finally, the optional tasks of calibration with a planar pattern and logo detection and insertion, are described in section 5 and section 6, respectively.

2 Robust Normalized DLT Algorithm

2.1 DLT explanation

The Direct Linear Transform (DLT) algorithm estimates the homography that relates two sets of point correspondences between two images. More concretely, the DLT algorithm estimates an homography H so that $x_i = Hx_i'$ for a set of correspondences $x \leftrightarrow x'$, where $x \in \mathbb{P}^2$ and $x' \in \mathbb{P}^2$. Therefore, it is important to first know that two sets of points x and x' from images I and I' are related by a homography. There are four cases for which this is true:

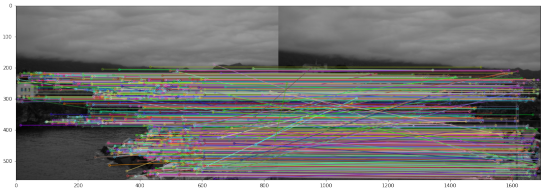
- The set of 3D points X which are imaged into the set of correspondences x and x' belong to the same 3D plane in the 3D world.
- The two images were obtained by rotating the camera about its center.
- The two images were obtained by just modifying the focal length.
- When the scene is far away from the camera, it can be considered that points belong to the plane at infinity $\sqcap_\infty$. Therefore, the two images are related by a homography, as we end up in the first case.

In the DLT algorithm, the equation $x_i = Hx_i'$ might be expressed as $x_i \times Hx_i' = 0$. This formulation allows the isolation of the unknown H in a linear system of the form $A_i h = 0$, where A_i contains elements of the known correspondence x_i and x_i' and h is the vectorized H. Only 4 point correspondences are needed to determine h, as each A_i contains two rows and with 4 correspondences we obtain a matrix A of rank 8. But as they might contain noise, it is more appropriate to use more point correspondences if provided. Due to noise, when we have more than four correspondences, the nullspace of A might be empty. Therefore, it is appropriate to solve the following minimization problem:

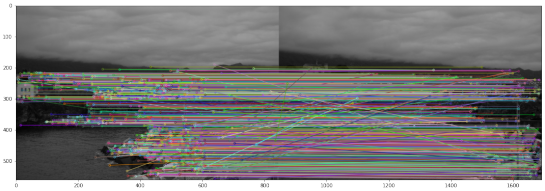
$$\begin{aligned} \min_h \quad & ||Ah||_2 \\ \text{subject to} \quad & ||h|| = 1 \end{aligned} \tag{1}$$

Nevertheless, this version of the DLT algorithm has two main problems:

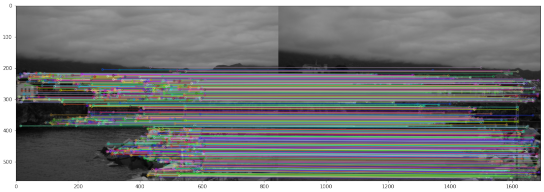
- It is not independent of the reference coordinate system of the points and this produces a greater variance in the solution under noisy points.
- As point correspondences are estimated in an automatic manner, there will be outliers in the correspondences. Outliers greatly affect the resulting homography.



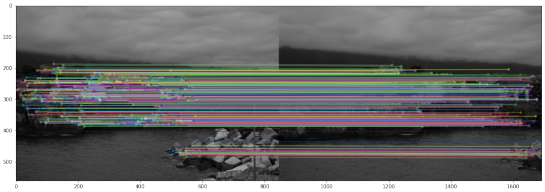
(a) Original correspondences between `llanes_a.jpg` and `llanes_b.jpg`.



(b) Original correspondences between `llanes_b.jpg` and `llanes_c.jpg`.



(c) Inlier correspondences between `llanes_a.jpg` and `llanes_b.jpg`.



(d) Inlier correspondences between `llanes_b.jpg` and `llanes_c.jpg`.

Fig. 1: Point correspondences before and after computing a homography with the robust DLT algorithm and keeping the inliers. In figures (a) and (b), we can observe the original point correspondences, which clearly contain outliers. In figures (c) and (d), only the inliers are kept after estimating the homography that relates the two images, which removes all visible outliers.

To solve the first problem, the normalized DLT algorithm is used, which normalizes points so that they are centered at the origin of the coordinate frame and their average distance to the origin is $\sqrt{2}$. For the second problem, RANSAC can be used to estimate a homography H with a minimal set of points and keep the H that produces the most number of inliers.

2.2 DLT: Implementation and results

The robust DLT algorithm is already implemented in the function *Ransac_DLT_homography* in the file *utils.py*. Inside this function, 4 random correspondences are selected to estimate H with the normalized DLT algorithm and the inliers are computed. The function *DLT_homography* uses the normalized DLT algorithm to compute H. The first step is to compute a normalizing transformation T for the set of points x and T' for the set of points x'. We want the centroid of the normalized points to be at $c = [0, 0]^T$ and the average distance to the centroid to be $\sqrt{2}$. To do so, we can construct the transformation T as:

$$T = \begin{bmatrix} s & 0 & -sc_x \\ 0 & s & -sc_y \\ 0 & 0 & 1 \end{bmatrix}$$

where the mean of the set of points x is $c = [c_x, c_y]^T$ and s is the average distance to the centroid, which can be computed as:

$$s = \frac{\sqrt{2}}{\frac{1}{N}\sum_{i=1}^{N} ||x_i - c||_2}$$

where N denotes the size of the set of points x. We can proceed in a similar way to obtain T'. Applying T and T' to x and x', respectively, we obtain a set of normalized correspondences which can be used in the DLT algorithm. We can construct from the set of correspondences matrices A_i that can be stacked into a matrix A to obtain the solution to the minimization problem of Equation 1. If we define the elements of x_i as $x_i = [p_1, p_2, p_3]$ and the elements of x'_i as $x'_i = [p'_1, p'_2, p'_3]$, each A_i is of the form:

$$A_i = \begin{bmatrix} 0^T & -p'_3 x^T & p_2 x^T \\ p'_3 x^T & 0^T & p_1 x^T \end{bmatrix}$$

Fig. 2: Mosaic of the set of images *"llanes"* obtained by using `llanes_b.png` as the reference image.

resulting in a 2×9 matrix A_i. The resulting homography $\hat{H}$ is found as the last column of V, where V comes from the SVD decomposition of $A = UDV^T$. As this matrix H has been estimated with the normalized set of correspondences, we need to obtain the denormalized H as $H = T'^{-1}\hat{H}T$. We can see that this is the correct denormalization as when applying the homography H to x_i, first x_i is normalized by applying T. Then, the homography estimated in normalized coordinates is applied, to obtain a point in the normalized coordinate system of I'. After which we apply T'^{-1} to obtain the point in the original coordinate frame of image I'.

In order to complete the robust DLT algorithm, the only task left is to define a function *Inliers* that decides which correspondences are inliers according to a proposed homography H. Given two correspondences x_i and x'_i, we decide to use the symmetric transfer error to determine whether this correspondence is an inlier given H. The symmetric transfer error d_t for each correspondence is defined as:

$$d_t = \sqrt{||[x'_i] - [Hx_i]||_2^2 + ||[x_i] - [H^{-1}x'_i]||_2^2}$$

where $[p]$ denotes the transformation of a point $p \in \mathbb{P}^2$ to $\mathbb{R}^2$. Then, one correspondence is an inlier if $d_t < t$, where t is a threshold.

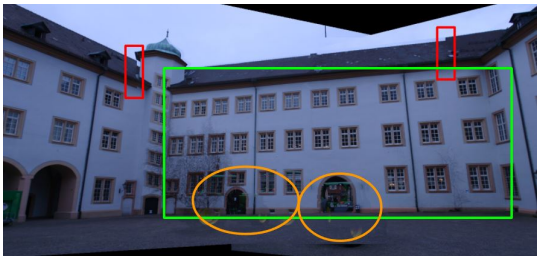
In Figure 1, we can observe the original correspondences between images and the result of only keeping the inliers according to the homography that relates the two images using the robust DLT algorithm.

3 Mosaics

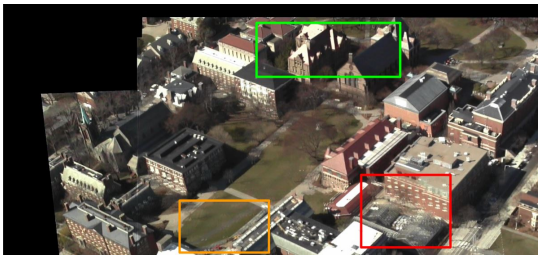
In this section, the robust DLT algorithm is applied to the creation of image mosaics. An image mosaic is created by stitching images to create one big image. If we establish point correspondences (SIFT + KNN matching, applying ratio test) between image pairs and compute the homography that relates them, we can transform images into the coordinate frame of one reference image and create the mosaic.

For doing so, we first select one reference image, for which we do not need to apply any homography. Then, we can compute the homography that relates other images to this reference image. By applying the estimated homography to images and blending them, we obtain the result in Figure 2 for the set of images *"llanes"*. Points in images seem to be related by a homography, as there are a lot of inlier correspondences, from the data we hypothesize that this is because images are taken by rotating the camera about its center.

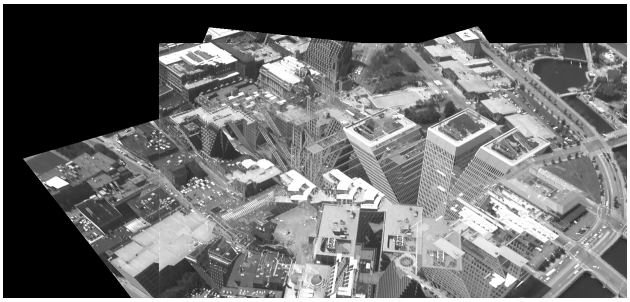
We compute mosaics for the set of images *castle_int*, *aerial/site13*, and *aerial/site22*. For the set *castle_int*, in Figure 3a, we can see that there is a translation between images, so the camera is not only rotating about its center. Based on the point correspondences, we can deduce that the homography relates correspondences in the front facade on the castle (indicated with a green rectangle in the figure). It produces a decent result in the part of the front facade because points in that facade belong approximately to the same 3D plane, hence they are related by a homography in the different images. In the left tower of the front facade, we can clearly see a region where the two images do not blend well (indicated with a red rectangle),



(a) Mosaic using the set of images *castle_int*.



(b) Mosaic using the set of images *aerial/site13*.



(c) Mosaic using the set of images *aerial/site22*.

Fig. 3: Mosaics created from different set of images.

as those points in the left facade do not belong to the front facade, creating artifacts. The same happens with smaller artifacts near the right facade. Objects which do not belong to that facade like the green truck (indicated with an orange circle) and the tree, are not well combined as they do not belong to the 3D plane of the front facade.

For the set *aerial/site13*, the resulting mosaic is depicted in Figure 3b. We can see from the inlier correspondences that the homography is estimated approximately from points of a plane at a certain height, relating points at the roof of buildings (e.g., the part marked with a green rectangle). At other places, such as the side of buildings we can see that the correspondences are not accurate (e.g., the part marked with a red rectangle). Furthermore, parts in the ground plane also seem to contain errors, for example, we can see how a road from another image is being blended in the floor (marked with an orange rectangle).

For the set *aerial/site22*, the resulting mosaic is depicted in Figure 3c. In this example, it is harder to obtain a homography and this results in many more iterations of the RANSAC algorithm. In this image, none of the assumptions mentioned in subsection 2.1 are true. The homography that is computed mainly relates points on the ground plane, which are the only ones that are more or less well fused in the mosaic. All other points, especially those belonging to the side of the tall buildings are not well fused, as they are not related by the estimated homography. Due to the number of tall buildings, the fusion is not good.

4 Refinement of the estimated homography with the Gold Standard algorithm

The measure we are minimizing in section 2 is the Algebraic distance. This measure does not have any relevant meaning geometrically for most transformations. There is another measure which does have this meaning, the Geometric distance: given correspondences between images, we can compute a distance measure (such as Euclidean) between the ground truth positions and the corresponding transformed positions from the other image.

$$\sum_{i=1}^{n} d(x_i, \hat{x}_i)^2 + d(x', \hat{x}'_i)^2 \quad \text{s.t. } \hat{x}'_i = H\hat{x}_i, \forall i \tag{2}$$

$$d(a, b) = (a_1/a_3 - b_1/b_3)^2 + (a_2/a_3 - b_2/b_3)^2 \tag{3}$$

On Equation 2, we can see the sum of geometry distances for a set of points. Note that it is symmetric, with a term for every direction of the homography. We call this sum the *reprojection error* when we have pairs

of perfectly matched points $\hat{x}'_i = H\hat{x}_i$. It can be shown that minimizing this error is equivalent to finding the Maximum Likelihood estimate of H (Hartley & Zisserman 4.3).

The Gold Standard algorithm uses this distance to find more meaningful homographies. We optimize both a set of subsidiary variables $\hat{x}$ and the homography on Equation 2. This algorithm is iterative, so it is recommended to initialize the homography using DLT for example, and to set the subsidiary variables to the measured x.

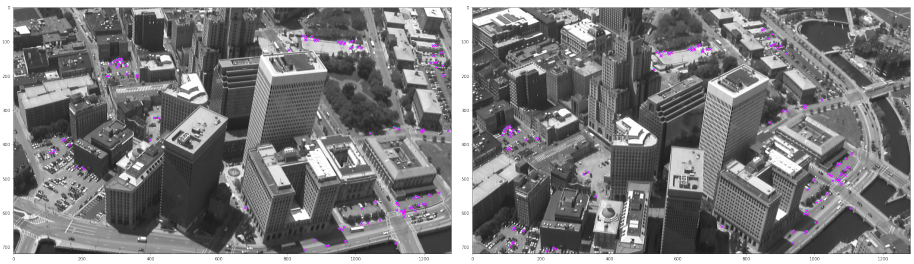
4.1 Implementation

We start by finding an initial estimate of the homography using DLT with RANSAC. We will use the set of inliers found for the Gold Standard algorithm: we set the initial subsidiary variables as $\hat{x}_0 = x$, and the transformed points will be $\hat{x}' = H\hat{x}$, where H will be our estimated homography.

In total, we have $2n + 9$ variables in our optimization process: the n 2-dimensional points (the homogeneous coordinate for the variable $\hat{x}$ will always be 1), and the 9 variables for H. We want to minimize Equation 2, and as we are using Levenberg-Marquardt with `scipy.optimize.least_squares` we will need more residuals than variables. This can be done by minimizing the 4 components of Equations 2 and 3 separately, as this is equivalent of optimizing the whole reprojection error. We should also note that the `scipy` function implicitly squares and sums the residuals.

For the homography from image 1 to 2 of the `aerial/site22` images, we reduce the geometric distance by half using the Gold Standard algorithm instead of the DLT: from a summed value of the symmetric distance of almost 100 for the inlier points, to 50. We can see on Figure 4 how the refined subordinate points overlap the correspondances we chose as *true*. Still, this set of images does not mosaic in a *clean* way: we hypothesise that this is because the two images are not related by a projective transform, as we can see by comparing the multiple views of the scene such as in Figure 4.

For other sets of images, such as the `llanes` one, we have seen a similar reduction of the already low geometric distance.



(a) Image *aerial_site22_1*. (b) Image *aerial_site22_18*.

Fig. 4: Two images with the ground truth and predicted/refined keypoints. We can see that they are practically overlapping.

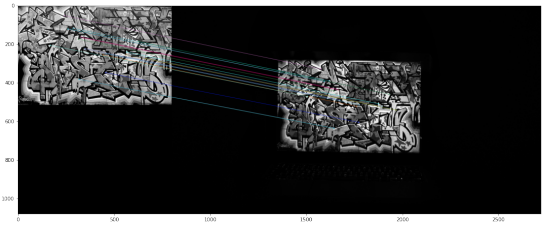
5 (Optional) Application: Calibration with a planar pattern and augmented reality

The goal of this section is to find the camera matrix $P_{3\times4}$ that models a camera and projects points from the real world into the image plane. This matrix can be decomposed as in Equation 4, where K is the calibration matrix (which contains the internal parameters of the camera) and R and t are the camera pose (and contain the external parameters of the camera).

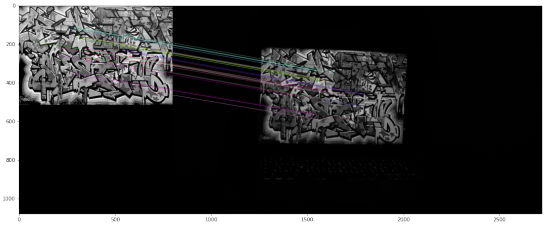
$$P = K[R|t] = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix} \tag{4}$$

In particular, we will use a calibration object (the planar pattern in Figure 5) to establish a set of 3D to 2D point correspondences from some views of this pattern to the planar pattern itself.

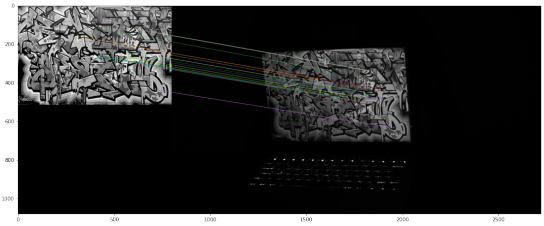
Fig. 5: Planar pattern used to calibrate the camera.



(a) Correspondences using the view *graffiti1*.



(b) Correspondences using the view *graffiti2*.



(c) Correspondences using the view *graffiti3*.

Fig. 6: Point correspondences between the views and the planar pattern.

5.1 Camera calibration

In order to calibrate a camera from a planar pattern we have to do the following steps:

1. **Estimate homographies:** First of all, we have to estimate the n homographies that relate the planar pattern and the n images or views of the pattern. This can be done using the Robust Normalized DLT algorithm previously implemented.
   In the example from the notebook, we have used 3 views. Their point correspondences with the planar pattern can be found in Figure 6.
2. **Build matrix V:** using the relation between a flat object and its image projection, and the projection to the image of a flat object at plane $Z = 0$, we get that $[\boldsymbol{h_1}\ \boldsymbol{h_2}\ \boldsymbol{h_3}] \sim K[\boldsymbol{r_1}\ \boldsymbol{r_2}\ \boldsymbol{t}]$. After some calculations we get the system of equations in Equation 5. Since we have 2 equations for 6 unknowns (the 6 entries in Ω) we need $n = 3$ views (homographies) to solve the system and find Ω. Therefore, the stack of these 2 equations for the 3 views leads to the $V_{2n\times 6}$ matrix. And, then, the system of equations to be solved is $V\Omega = 0$.

$$\begin{cases} \boldsymbol{V}_{12}^T\Omega = 0 \\ (\boldsymbol{V}_{11}^T - \boldsymbol{V}_{22}^T)\Omega = 0 \end{cases}, \text{ where } \begin{array}{l} \boldsymbol{V}_{ij}^T = (h_{1i}h_{1j}, h_{1i}h_{2j} + h_{2i}h_{1j}, h_{1i}h_{3j} + h_{3i}h_{1j}, \\ \qquad\qquad h_{2i}h_{2j}, h_{2i}h_{3j} + h_{3i}h_{2j}, h_{3i}h_{3j}) \end{array} \tag{5}$$

3. **Find ω:** Minimizing $\|V\Omega\|_2$ under the condition that $\|\Omega\|_2 = 1$, we find the best solution for $V\Omega = 0$. This is equivalent to computing the singular vector decomposition of $V = UD\bar{U}^T$ and taking as Ω the singular vector associated to the smallest singular value (last column of $\bar{U}$).
Rearranging the values of $\Omega = (\omega_{11}, \omega_{12}, \omega_{13}, \omega_{22}, \omega_{23}, \omega_{33})^T$, we get:

$$\omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{bmatrix} = \begin{bmatrix} 1.00 \cdot 10^{-7} & -8.62 \cdot 10^{-11} & -7.15 \cdot 10^{-5} \\ -8.62 \cdot 10^{-11} & 1.02 \cdot 10^{-7} & -8.85 \cdot 10^{-5} \\ -7.15 \cdot 10^{-5} & -8.85 \cdot 10^{-5} & 1.00 \end{bmatrix}$$

4. **Extract calibration parameters, K:** By definition we know that $\omega = K^{-T}K^{-1}$. We also know that K is an upper triangular matrix, so is K^{-1}. Therefore, K^T and K^{-T} are lower triangular matrices. Since we have the product of a lower triangular matrix (K^{-T}) with an upper triangular matrix (K^{-1}), we can use the Cholesky decomposition to obtain K^{-1}. Then we just have to invert it to get K. Since we want a value 1 in K_{33}, we also divide K by K_{33}.
In the example from the notebook, the matrix of internal parameters is:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2945.34 & 2.50 & 711.85 \\ 0 & 2918.93 & 865.00 \\ 0 & 0 & 1 \end{bmatrix}$$

where s is the skew, which is usually 0, so our value 2.5 is very close. $(x_0, x_y)^T$ are the coordinates of the principal point in pixels, which would be ideally in the center of the image; hence, the bigger the image, the bigger are x_0 and y_0. In this case, the images are 1080×1920, so ideally $(x_0, x_y) = (540, 960)$ which is close to our values $(711.85, 865.00)$. α_x and α_y represent the focal length of the camera in terms of pixel dimensions in the x and y direction respectively. Ideally, pixels would be square, $\alpha_x = \alpha_y$. In our case, pixels are close to square since $\alpha_x = 2945.34 \approx 2918.93 = \alpha_y$.
5. **Extract R and t:** Using $[\boldsymbol{h_1}\ \boldsymbol{h_2}\ \boldsymbol{h_3}] \sim K[\boldsymbol{r_1}\ \boldsymbol{r_2}\ \boldsymbol{t}]$ and since $\|r_1\| = \|r_2\| = 1$ and r_1, r_2, r_3 are orthogonal, we get the formulas to find R and t:

$$r_1 = \frac{K^{-1}h_1}{\|K^{-1}h_1\|}, \qquad r_2 = \frac{K^{-1}h_2}{\|K^{-1}h_2\|}, \qquad r_3 = r_1 \times r_2, \qquad t = \frac{K^{-1}h_3}{\|K^{-1}h_1\|}$$

But, to guarantee that R is an orthogonal matrix, we impose it:

$$\tilde{R} = U\bar{U}^T, \text{ where } R = UD\bar{U}^T.$$

Drawings of the planar pattern and the camera locations In Figure 7 we have two equivalent ways to represent the relative position of the 3 views of the planar pattern.

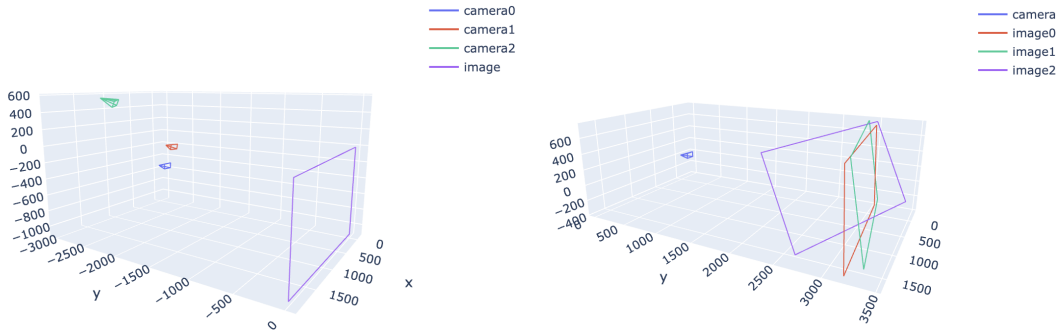
In Figure 7a we can see the drawing of the planar pattern and the 3 camera locations (one for each view). To create this plot, we used the functions `plot_image_origin()` and `plot_camera()`, which uses the function `optical_center()` to compute the optical center of the camera. The camera projection centre, C, is the only point for which the projection is not defined. That is, $P\begin{bmatrix} C \\ 1 \end{bmatrix} = 0$. So, we have to find C that minimizes $\|P\begin{bmatrix} C \\ 1 \end{bmatrix}\|_2$. This is equivalent to computing the SVD of $P = UDV^T$ and taking as $\begin{bmatrix} C \\ 1 \end{bmatrix}$ the singular vector associated to the smallest singular value (last column of V) normalized by its last value.

In Figure 7b we can see the drawing of a fix camera and the 3 planar patterns at different relative poses with respect to the camera. To find the position of the corners of the planar patterns, we have to multiply each image corner by the corresponding pose matrix:

$$[R\ t] = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

5.2 Augmented reality

To insert a cube on top of the different views of the planar pattern:



(a) Drawing of the planar pattern and the 3 camera locations.

(b) Drawing of a fix camera and 3 planar patterns at different relative poses w.r.t. the camera.

Fig. 7: Drawings of the planar pattern and the camera position.

- We first have to define the corners of the cube, in the range $0, 1$, for example.
- Secondly, we compute the *offset* as the center of the template (where we want our cube to be placed).
- Third step is to center the previously defined corners at zero, make the size of the cube (its edges) a quarter of the width of the template, and finally add the previously computed offset to each corner in order to center the cube at the center of the template.
- Also, we add a 1 to each corner point, in order to have them in homogeneous coordinates $\mathbb{P}^4$.
- Then, for each view, we multiply the corner points by the view's projection matrix P, to obtain their projection on the image plane in homogeneous coordinates $\mathbb{P}^2$; so we have to divide by the last component to get a 1 at that position.
- Finally, we just have to read and plot the image, and draw the corresponding lines between the corners to obtain a cube.

As we have already seen, the projection matrix P has two components: the extrinsic parameters, R and t, and the intrinsic parameters, K. The first ones are the camera pose, and define the location and orientation of the camera with respect to the world frame. The second ones are the calibration matrix, and allow a mapping between camera coordinates and pixel coordinates in the image frame. Therefore, the camera view needs to be calibrated in order to get the final coordinates of the corners into the image view. Without the camera calibration matrix the cube doesn't even appear in the image (it goes beyond the image frame).

The results can be seen in Figure 8, with the three views used to find the camera calibration in subsection 5.1.

6 (Optional) Logo detection and insertion

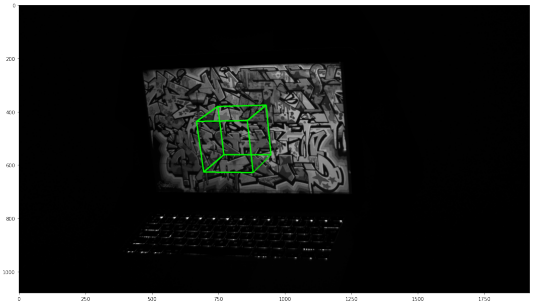
Logo detection and insertion is performed by using solution from previous chapters. First, we find matches between the logo we want to detect and where the target image contains the logo. By finding the matches we are able to estimate the homography between the plane logo and the target image. Then, we map corners using the estimated homography.

Once the logo is found on the target image and we have estimated the corners of the logo, we apply the homography to the new logo and create the mask that is used for replacement of the previous logo on the target image. Final result can be seen on Figure 9 also we can see the example when the detection fail to find a logo and creates wrong homography in Figure 10.

During the testing of the solution with newly taken pictures of the UPF logo, we have noticed that the automatic placement is highly sensitive to the SIFT descriptor finding right keypoints. Proving that it is probably not the best option to detect logo to be replaced and better option would be to first detect the corners of the logo to be replaced and then estimate homography from better detected corners using different method.

(a) Virtual object on the view *graffiti1*.

(b) Virtual object on the view *graffiti2*.



(c) Virtual object on the view *graffiti3*.

Fig. 8: Drawing of a 3D virtual object (cube) on top of the different views of the planar pattern.

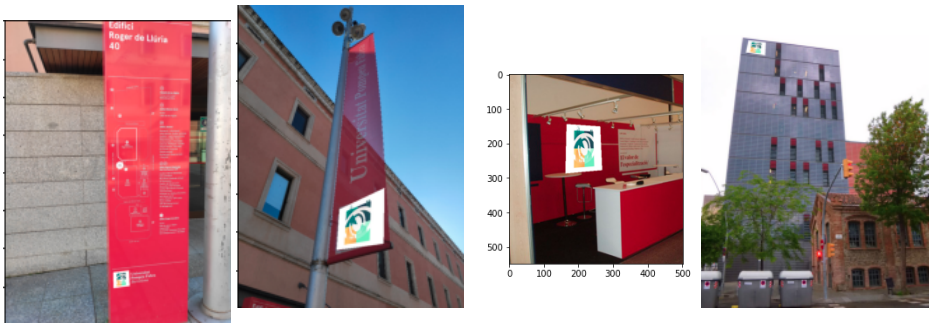


Fig. 9: Successful cases of logo detection and placement

Fig. 10: Failed case

7 Conclusions

In this week's assignment, we have seen that it is important to know if two images are related by a homography, otherwise only some points will be somewhat related. This will result in a mosaic that will not be what we expect. One important application of the estimated homography is that if it is estimated using the robust DLT algorithm, it can be used to remove outlier correspondences. Furthermore, we have seen how

minimizing geometric errors such as the reprojection error can lead us to more meaningful and expected results. On the optional task of camera calibration using a planar pattern, we have seen how the camera calibration matrix plays an essential role in the insertion of an object on top of an image of a flat pattern. Without it, we cannot map from camera coordinates to pixel coordinates and the object is not placed in the right position.