Recovery of 3D Urban Scenes: Homographies for Panorama Reconstruction, Camera Calibration, and Logo Replacement

Álvaro Budria, Alex Carrillo, Sergi Masip and Adrià Molina

Universitat Pompeu Fabra

January 2022

1 Introduction

In this document, we present our results after implementing and testing two approaches for homography estimation: the robust normalized direct linear transform, and the Gold-Standard algorithm. In addition to that, we implement and test Zhang's method for camera calibration, and develop a system for automatically detecting and replacing logos.

1.1 When does a homography relate a pair of images?

If we know the homography transformation that relates two images, we are then able to compute the corresponding points from one image to the other, i.e. given points x_i and x_i', and homography H relating them, we have that $x_i' = Hx_i$ and $x_i = H^{-1}x_i'$.

Understanding when we may or may not relate images via a homography is key to our analysis of results in Sec. 2.3. Briefly stated, two images may be related by a homography when

- all the imaged points lie on the same 3D world plane;
- the image is taken far away from the imaged points, in which case they may be assumed to lie on the same plane;
- the images are taken with a static camera with varying focal lengths;
- the images are taken with a camera rotating about its centre.

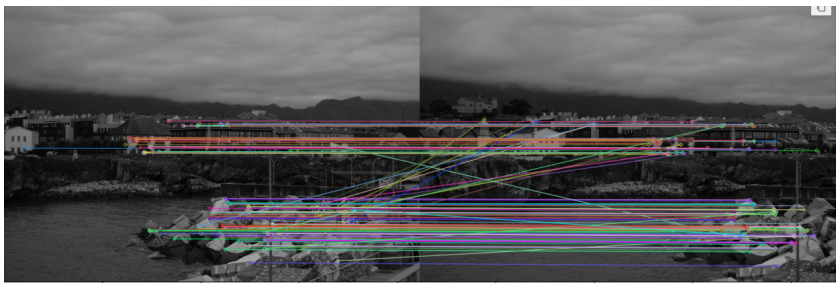
If none of these conditions is met for a given pair of images, then we will invariably fail to estimate a homography relating them.

2 Homography estimation with the DLT algorithm

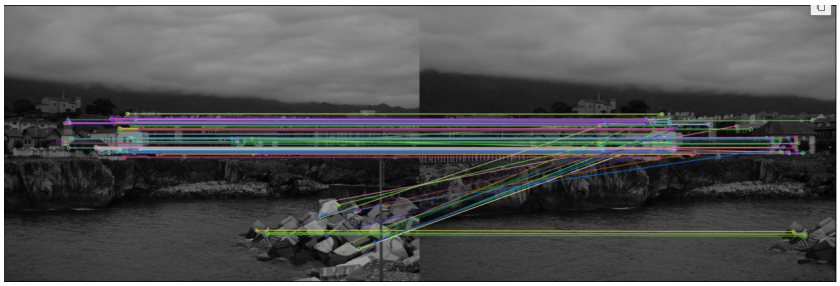
In this section, we estimate the homography that relates two images with the robust normalized direct linear transform algorithm (DLT).

2.1 Obtaining point correspondences

The DLT algorithm relies on the availability of at least $n = 4$ correspondences between the two images. For this reason, we first generate 2D point correspondences between the images. We detect keypoints in both images with ORB. Then, we match those keypoints with a fast kNN matcher, thus obtaining the correspondences. At this point, we filter out the least confident matches with Lowe's ratio test. Fig. 1 illustrates this keypoint matching between different viewpoints of the same scene. It is obvious that in this particular example, there are a few mismatches, as some correspondence lines cross each other. These errors would cause the DLT algorithm to fail to capture the relation between the images. Nevertheless, we do not filter these errors at this point, and trust RANSAC to take care of them.

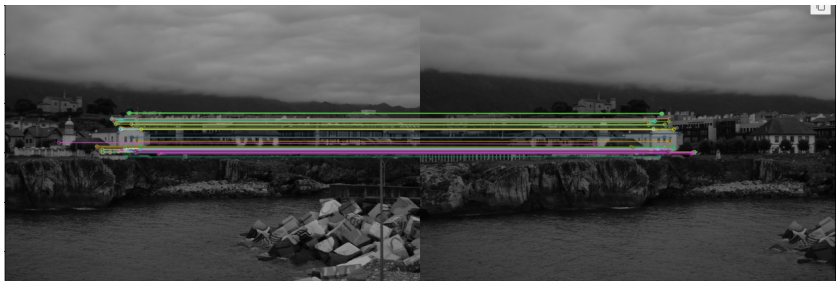


(a) Initial matches, images 1 and 2.



(b) Initial matches, images 2 and 3.

(c) RANSAC-filtered matches, images 1 and 2.



(d) RANSAC-filtered matches, images 2 and 3.

Figure 1: Image Correspondences in the *Llanes* scene.

2.2 Computing the homography for a pair of images

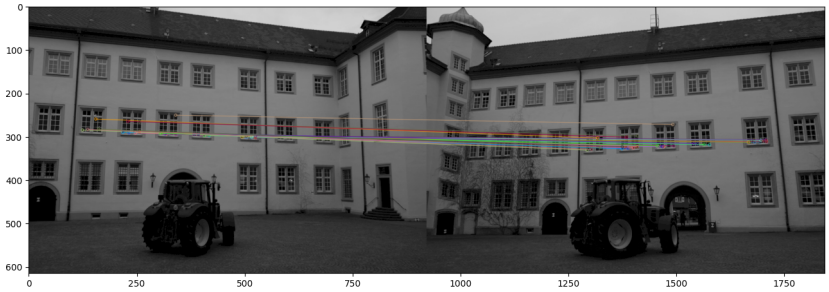
Given the 2D correspondences, we employ the normalized DLT algorithm robustified with RANSAC, in order to estimate the relating homography H.

The result of vanilla DLT is dependent on the choice of coordinate frame for the input points. To make it more stable, it is a good practice to normalize the points previous to computing the homography and then denormalise the estimated homography. In this normalization, we centre the centroid of points at the origin and scale their average distance to $\sqrt{2}$.

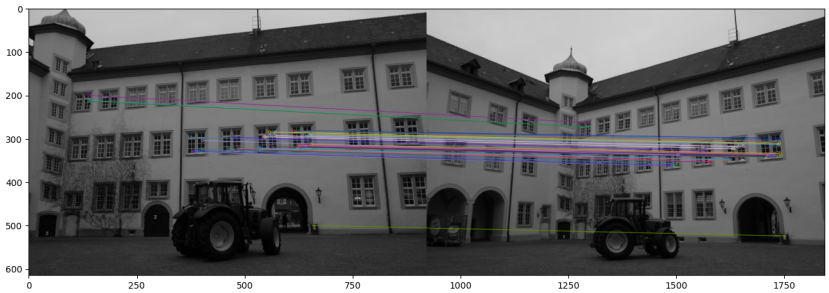
By incorporating RANSAC, we make the whole process most robust to erroneous matches, such as those that can be seen in Fig. 1 (a, b). At each iteration, RANSAC randomly samples 4 correspondences (as we need a minimum of 4 to get the necessary constraints), computes a homography, and counts the number of inliers (correspondences having an error below a threshold t). Additionally, we also reestimate at each iteration the maximum number of iterations, based on the proportion of inliers. In Fig. 1 (c, d) we can see those matches that are selected as inliers, from which we compute the final homography H. Notice how the mismatches have withered away, proving that RANSAC successfully selects the homography that best captures the relation between two images, or at least on that is very good.

As an additional experiment, we played with the threshold for the inliers. We tested on the *Castle* scene, which is

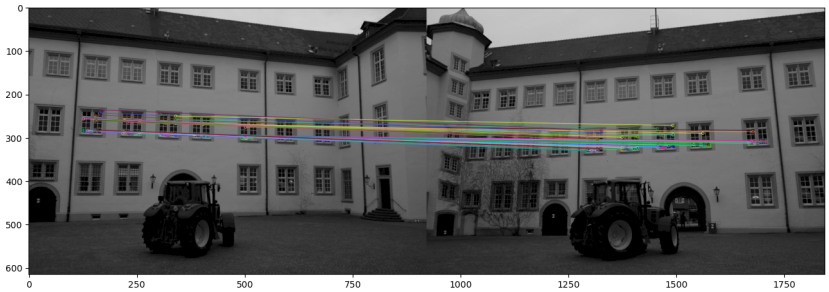
interesting because all of the similar windows lie on the same planar surface. These make the keypoint matcher yield plenty of matches. We notice the larger the threshold is, the faster the algorithm is. However, it also filters fewer matches, thus leaving some lower-quality ones (such as those from objects that do not lie on the same planar surface, see Fig. 2). Therefore, the improvement in performance comes at the cost of the quality of the final mosaic. In the mosaic generated with $th = 32$ (Fig. 3) we can see that there are more defects and misalignments than in the mosaic generated with $th = 3$ (Fig. 5). In contrast, if we set a threshold too low (such as $th = 1$), the algorithm will take too long and will not stop until it reaches the maximum iterations.



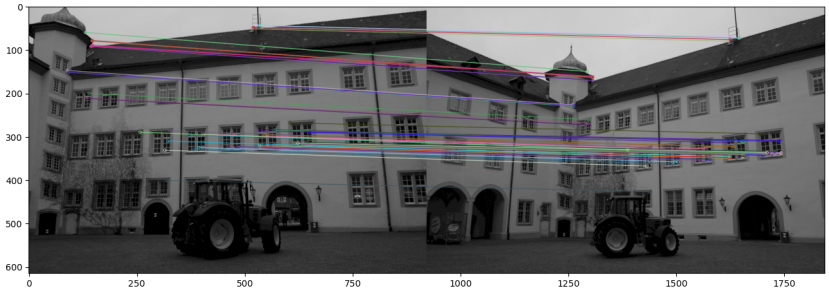
(a) RANSAC-filtered matches (th=3), images 1 and 2.



(b) RANSAC-filtered matches (th=3), images 2 and 3.



(c) RANSAC-filtered matches (th=32), images 1 and 2.



(d) RANSAC-filtered matches (th=32), images 2 and 3.

Figure 2: Image Correspondences in the *Castle* scene with different inliers threshold.

2.3 Building a panorama mosaic from a set of relating homographies

Given three images I_1, I_2, and I_3, as well as relating homographies $H_{1,2}$ and $H_{2,3}$, we are able to fuse the images into a single mosaic as a panorama view. To do so, first we transform images I_1 and I_3 into the same frame as I_2: $\hat{I_1} = H_{1,2}I_1$,

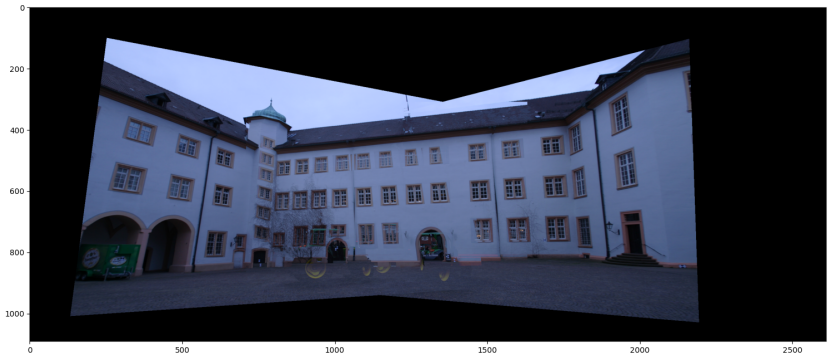


Figure 3: **Mosaic of the *Castle* scene with threshold for the inliers set to 32.**

$\hat{I}_3 = H_{2,3}^{-1} I_3$. Then, we can just plot them onto the same single canvas having an appropriate size to fit all three images. Note we are assuming prior knowledge about the relative location of the camera with which the images were taken. We need to know which image is left, which one is center, and which one is right.

In our experiments, we compute relating homographies from four different image sets to build a mosaic.

The first mosaic we make is for the *Llanes* scene. We assume it to be planar, and that all points we are considering lie on the same real 3D world plane. Since the three images were taken with a static camera rotating about its centre and the picture is taken from afar, this assumption is most likely valid. The good results obtained (see Fig. 4) indicate that the assumption holds.

Figure 4: **Mosaic of the *Llanes* scene.** Notice the double funnel shape, which is common when building a mosaic from images taken rotating the camera.

Still reasonably good, though not as seamless as in *Llanes*, the second mosaic is made of the set of images of the *Castle* scene (see Fig. 5). In this case, the camera does not seem to be fixed. The most noticeable defect in this mosaic is the "phantom" tractor. The reason for the tractor's disappearance may be that it does not lie on the same plane in the real 3D world as the facade points. This violates a key assumption for relating points in different 2D images, namely, that points lie on the sample plane.

Despite the tractor's failure, the rest of the scene is more or less adequately fused, because, actually, all of the matched keypoints used to set up the correspondences are from the facade.

In this scene, we increased the distance threshold in the ratio test in order to obtain more initial matches. Still, RANSAC got rid of the spurious matches on the tractor.

As in the first scene, the three pictures from *Aerial/site13* are taken from a distance. Moreover, this time the pictures are taken from a very high angle, which strengthens the assumption that the photographed points lie on the same plane (to see why, consider that the vertical lines would look like dots when viewed exactly from above: their apparent length is the length multiplied by the cosine of the elevation angle). Despite the buildings having a certain height, the effect of the angle and the fact that they are not very tall does not hinder the planar assumption (see Fig. 6).

In scene *Aerial/site22*, the DLT algorithm has obviously failed to fuse the views into a single panorama (see Fig. 7). This is due to the fact that the planar assumption is violated. The scene contains buildings of very different heights: regular buildings and skyscrapers. The geometry of the scene is not planar, and thus different views of it cannot be related by a homography. In addition to that, the small number of correspondences may also be a cause of this failure from DLT to compute the homography (after all, since the points are not coplanar, viewing them from different angles leads to some of them being occluded or looking much differently, so it is harder to match them).

Figure 5: **Mosaic of the *Castle* scene.**

Figure 6: **Mosaic of the *Aerial/site13* scene.**

3 Refining homography estimates using the Gold Standard Algorithm

In this section, we are trying to make the estimated homographies more accurate by using a method called the Gold Standard algorithm. In previous steps, we used the Algebraic distance as the measure. This measure does not have any relevant meaning geometrically for most transformations. Instead, there is another measure we can use which does have this meaning: the Geometric distance. Thus, given correspondences between images, we can compute a distance measure, such as Euclidean, between the real positions of points (ground truth) and the corresponding transformed positions from the other image.

$$\begin{aligned} &\sum_{i=1}^{n} d\left(x_i,\hat{x}_i\right)^2 + d\left(x',\hat{x}'_i\right)^2 \quad \text{s.t.} \quad \hat{x}'_i = H\hat{x}_i, \quad \forall i \\ &d(a,b) = \left(a_1/a_3 - b_1/b_3\right)^2 + \left(a_2/a_3 - b_2/b_3\right)^2 \end{aligned} \tag{1}$$

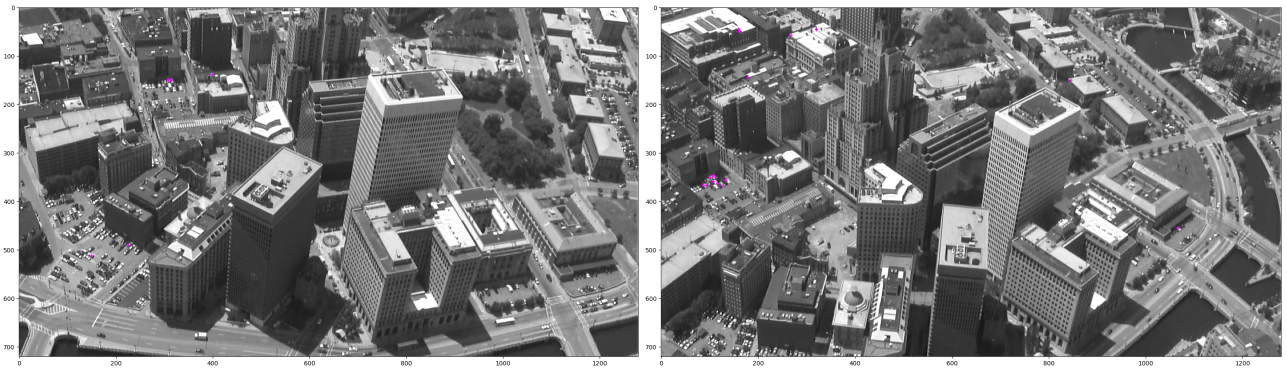
In Equation 1 the sum of geometric distances is defined for a set of points. Note that this sum is symmetric because it contains a term for each direction of the homography. This sum is called the *reprojection error* when the pairs of points are perfectly matched $\hat{x}'_i = H\hat{x}_i$. It can be demonstrated that minimizing this error is equivalent to finding the Maximum Likelihood estimate of H (as shown in Hartley and Zisserman, Sec. 4.3).

The Gold Standard method employs the distance presented in Equation 1 to determine more meaningful homographies. This algorithm optimizes both a set of auxiliary variables $\hat{x}$ and the homography to refine. And, as it is an iterative process, it is suitable to initialize the homography using DLT algorithm and use the measured x as the auxiliary variables.

More specifically, an initial estimate H of the homography is obtained through DLT with RANSAC. This estimate is then used in the Gold Standard method with the set of inliers found. The initial auxiliary variables are set as $\hat{x_0} = x$ and the transformed points are represented as $\hat{x}'_i = H\hat{x}_i$.

In fact, the Gold Standard approach simplifies to a non-constrained minimization problem on the objective function of Equation 1, with a total of $2n+9$ variables. These include n 2-dimensional points (as the homogeneous coordinate for x

Figure 7: **Mosaic of the *Aerial/site22* scene.**



(a) Keypoints of image 1 (`aerial/site22/01`)

(b) Keypoints of image 2 (`aerial/site22/18`)

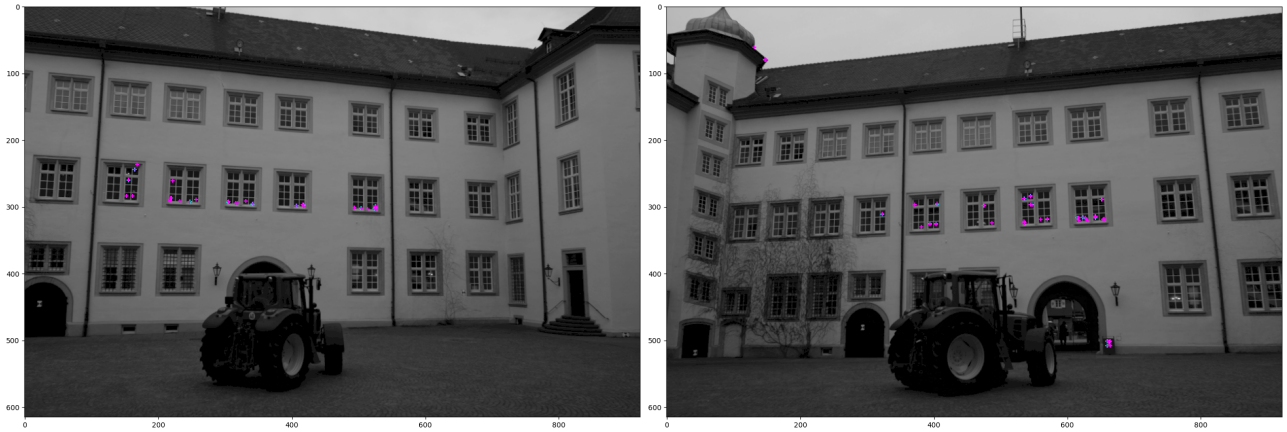
Figure 8: Visualization of the differences between the ground truth (cyan crosses) and predicted/refined keypoints (purple crosses) for `aerial/site22` images. The result of the Gold Standard method shows that points are essentially overlapping.

is always 1) and 9 variables for H. To solve this, we use the Levenberg-Marquardt algorithm using *least squares* function from `scipy`. To ensure that we have enough residuals, we minimize the 4 components of Equation 1 separately, as it is equivalent to optimizing the entire reprojection error. It is worth noting that squaring the terms is not needed as the least squares function automatically constructs the cost function as a sum of squares of the residuals.

	`aerial/site22/1`	`aerial/site22/18`	`castle_int/14`	`castle_int/15`
Geometric error before refinement	12966250.12	5015221.65	58882687.92	176516788.08
Geometric error after refinement	1.39	5.51	8.53	9.85

Table 1: Results of the geometric error before and after the refinement with the Gold Standard algorithm. The reduction of the error is widely achieved.

When performing the refinement of the estimated homography with the Gold Standard algorithm, we drastically reduce the geometric distance by several orders of magnitude when tested on both the `aerial/site22` and `castle_int` set of images. To better compare the two homographies (refined and non-refined), Table 1 shows a comparison of the geometric error on images before and after the refinement. As shown in Fig. 8 and Fig. 9, the refined points overlap well with the correspondences we identified as true. However, when we built the mosaic on both sets of images with the refined homographies, we get particular results. For the `aerial/site22` set, the images do not seamlessly combine in the mosaic of Fig. 10. But, as already commented in previous sections, this could be due to the fact that the aerial images are not related by a projective transform, as also seen when comparing multiple views of the scene in the previous mosaic of Fig. 7. We have also observed a similar reduction in the geometric distance for other sets of images, such as the `castle_int` set. In the case of this mosaic with refined homographies of Fig. 11, the results seem not as good as the ones obtained without refining. However, the mosaic is still more visually appealing in terms, for instance, of facade alignment.



(a) Keypoints of image 1 (`castle_int/14`)

(b) Keypoints of image 2 (`castle_int/15`)

Figure 9: Visualization of the differences between the ground truth (cyan crosses) and predicted/refined keypoints (purple crosses) for `castle_int` images. The result of the Gold Standard method shows that points are essentially overlapping.

Figure 10: Mosaic of the `aerial/site22` scene with refined homographies.

Figure 11: Mosaic of the `castle_int` scene with refined homographies.

4 Camera calibration from a planar pattern (Zhang's method)

We may calibrate a camera (i.e. estimate its intrinsic and extrinsic parameters) with a planar pattern with Zhang's algorithm.

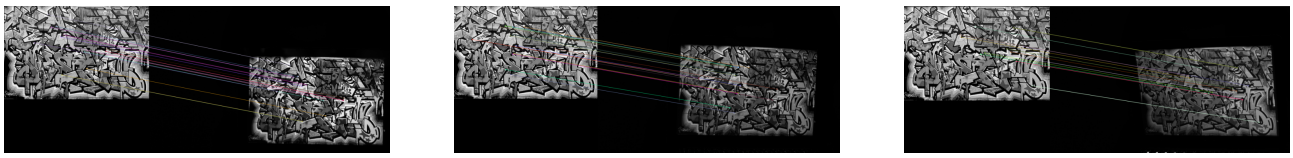


Figure 12: The matches between ORB keypoints in the template image and the three views of the planar pattern.

A key assumption behind Zhang's method is that the flat pattern is located at the $Z = 0$ plane. This allows us to ignore the third column r_3 of the rotation matrix $R = \begin{bmatrix} r_1 | r_2 | r_3 \end{bmatrix}$.

First, we relate each photograph with the template image via a homography. This is possible because all the imaged points on the pattern lie on a plane by the construction of this method. Note that all pictures must be taken with the same camera we want to calibrate.

Then, we set up $2 \cdot N = 6$ constraints, 2 per view:

$$\begin{cases} \mathbf{V_{12}^T} \Omega \\ \left(\mathbf{V_{11}^T} - \mathbf{V_{22}^T} \right) \Omega \end{cases} \tag{2}$$

and then we stack all these constraints together into a single matrix $\mathbf{V}$ of size 6×6. This allows us to compute the coefficients in Ω of the Image of the Absolute Conic ω. We solve for Ω by running an SVD on $\mathbf{V} = UD\bar{U}^T$ and taking the last column of $\bar{U}$. Then we rearrange the coefficients into a symmetric 3×3 matrix ω.

Given this $\omega = (KK^T) = K^{-T}K^{-1}$, we can use Cholesky decomposition to obtain K^{-1} and then obtain the inverse, K, which specifies the intrinsics of the camera. Note that ω must be positive definite in order to allow for a Cholesky decomposition. In some cases, due to randomness and numerical errors introduced during the estimation of the homographies, ω might not be positive definite. In our experiments, though we did not encounter this issue. All in all, the extrinsics can be estimated as

$$\begin{cases} r_1 = \frac{K^{-1}h_1}{||K^{-1}h_1||} \\ r_2 = \frac{K^{-1}h_2}{||K^{-1}h_2||} \\ r_3 = r_1 \times r_2 \\ t = \frac{K^{-1}h_3}{||K^{-1}h_1||} \end{cases} \tag{3}$$

In our case, the intrinsics we obtained are

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2.60 \cdot 10^3 & 2.35 & 9.10 \cdot 10^2 \\ 0.00 & 2.63 \cdot 10^3 & 5.04 \cdot 10^2 \\ 0.00 & 0.00 & 1.08 \end{bmatrix}. \tag{4}$$

where f_x and f_y are the focal lengths on the x and y axis, respectively, (x_0, y_0) is the principal point and s is the skew factor. In our case, the principal point is $(x_0, y_0) = (9.10 \cdot 10^2, 5.04 \cdot 10^2)$. Note how the focal lengths $f_x = 2.60 \cdot 10^3$ and $f_y = 2.63 \cdot 10^3$ differ in our estimation. We attribute this to numerical errors in the calibration process as well as focal distortion in the camera. The skew coefficient is $s = 2.35 \cdot 10^1$, quite far from 0, which is probably due to focal distortion when taking the pictures, which induces non-square pixels. This is a typical phenomenon when taking an image of an image, as is our case, where the axis of the lens is not parallel to the plane of the image.

Having obtained the extrinsics of our camera, we are now able to find the optical centre and guess the real-world location of the camera. We compute the optical centre via an SVD of P (the projection from 3D space to 2D space), $P = UDV^T$. The optical centre in homogeneous coordinates corresponds to the last singular value and is retrieved as the last column of V.

We can visualize camera positions with respect to the plane, or consider a fixed camera and three rotated planes:

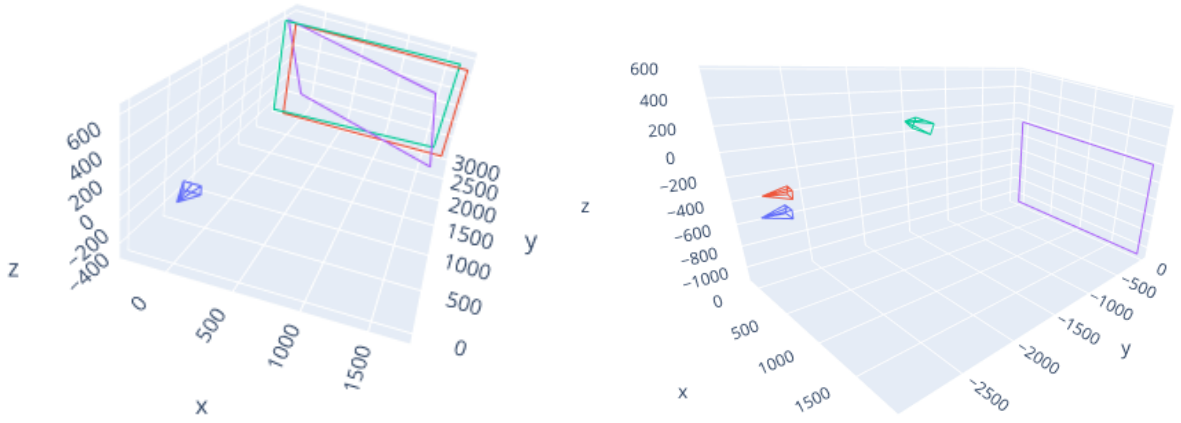


Figure 13: Pose of either three planes w.r.t. a fixed camera (left) or three cameras w.r.t. a fixed plane (right).

5 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 on top of the flat pattern in a way that is consistent, in a perspective sense, with the rest of the scene. For doing so, the following piece of `Python` code has been presented in the following algorithm:

1: $T_w, T_h \leftarrow C^{shape}_{pattern}$
2: `let` K ▷ Define an object K as a world-coordinates set of points
3: $K' \leftarrow \begin{bmatrix} K_{2xn} \\ 1_{1xn} \end{bmatrix}$ ▷ Convert to homogeneous coordinates x, y, z, μ
4: **for** $p_{view} \in V_{views}$ **do** ▷ For each projective matrix in the set of views
5: $\quad x_i \leftarrow p_{view} \times K'$ ▷ Get the planar coordinates for the world-points x', y', z', μ'
6: $\quad x_i \leftarrow x_i^{x',y',z'} / x_i^{\mu'}$ ▷ Project back to a surface
7: $\quad draw(x_i, Image_i)$ ▷ Although the code draws lines, simplified to drawing points

---

It's important to calibrate the camera for a simple reason: knowing its position and orientation. This allows us to create projective matrices that don't depend on any view or perspective; therefore we can directly create matrices from world coordinate into a certain perspective directly, so it's easy to define objects as a set of coordinates. Otherwise, for drawing the same object several times one should take one perspective as the "true" and do two different sets of points relative to this initial perspective, which is awkward and not feasible in actual AR applications where the perspective changes every second in an uncontrolled environment. Knowing the position of the camera allows us to create a single representation of an object and project it anywhere by just changing the transformation matrix, but not the representation of the object itself.

6 Logo detection and replacement

One of the proposed applications for the method is the replacement of logos in real-scene images. Logos are planar objects that suffer projective transformations and, therefore, replacing them while keeping a "natural" appearance is not as trivial as just applying a patch in the target image.

The first remark we have to take into account is the different nature of both the template (the logo) and the target (the scene) image. As the logo tends to be a small simple signal with a uniform style and, therefore, a low variety of features; the matching may be unstable and not robust at all Fig 18.

6.1 Logo RoI detection

In order to bypass the non-robustness shown in Fig. 18 we propose to consider only a detected Region of Interest (RoI) where the relevant keypoints will lay. In this matter, we convolved the original image with the logo as the kernel (performing a correlation) resulting in a good RoI for the keypoint matching as it's shown in Fig. 17.

The last step is to estimate the homography, which we expect to have a big translation vector $(t_x, t_y) = (h_{1,3}, h_{2,3})$ Eq 5

since it will be able to warp this small object (in relation to the rest of the building) into a "logo-like" image (Fig. 14).

$$H = \begin{bmatrix} 0.6 & -0.07 & 100 \\ 0.09 & 0.4 & 10 \\ 2 \cdot 10^{-4} & -1 \cdot 10^{-4} & 0.5 \end{bmatrix} \tag{5}$$

6.2 Logo replacement with estimated homography

For taking back the logo to its expected place, all we have to do is perform the warping with H^{-1}, which results as shown in Figs. 15 and 16.

Despite obtaining an adequate rectification of the images, the proposed methods have multiple limitations. It's important to recall that both metric and affine rectification need *ad-hoc* information in order to perform, being this information the segments used as reference. Because of this, the performance of the method in real-world applications is either limited to those circumstances where a human input can be expected or to the performance of the line detection algorithm alongside its own performance.

Nevertheless, in contrast to deep learning solutions, it provides a robust approach based on the foundations of image formation and geometry, rather than a probabilistic approach that returns likely or convincing outputs given an image.

7 Conclusions

In this practicum, we have explored the concept of homography estimation and how to relate pairs of images. In addition to that, we also implement Zhang's method for camera calibration. As an application of the theoretical insights, we tackle the problem of logo detection and removal.

Despite having obtained some good results when relating images via homographies, we failed to do so in some scenes, such as `aerial/site22`. Nevertheless, we attribute this failure to the fact that the captured scene is not planar, and therefore different views of it are not related via a homography.

With the Gold Standard algorithm, we successfully refine the homographies estimated for all image pairs, which enhances the quality of the reconstructed panoramas. Still though, those scenes violating the key assumptions stated in Sec1.1 are impossible to relate via a homography.

When it comes to augmented reality, it's important to remark that calibrating the camera allows us to create a single representation of an object (in this case, the cube) which for instance is very useful in real-world applications. Without this calibration, we should have a different representation of the same object for every possible perspective the user wants to get from the AR system, which is not feasible at all. Although some of them use tricks to bypass visual challenges like occlusions or patterns that are hard to calibrate with techniques like active illumination or stereo, AR software, many times, take advantage of knowing a world-coordinates system that allows us to have robust systems that don't fall apart to unexpected changes on the viewport while having a single representation for each projected object.

Regarding logo detection and replacement, we've shown the importance of a proper selection of the keypoints in order to build robust algorithms. Many times it's not possible to be solved through a strict keypoint detection approach but introducing some priors (such as the existence of a template) that allow you to bypass this matter.

All in all, this lab session has demonstrated some of the amazing applications that projective geometry has to offer, namely, homography estimation and camera calibration.

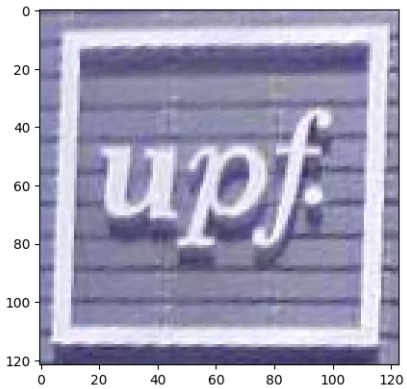


Figure 14: Resulting image for warping the target image with its estimated homography H (top).

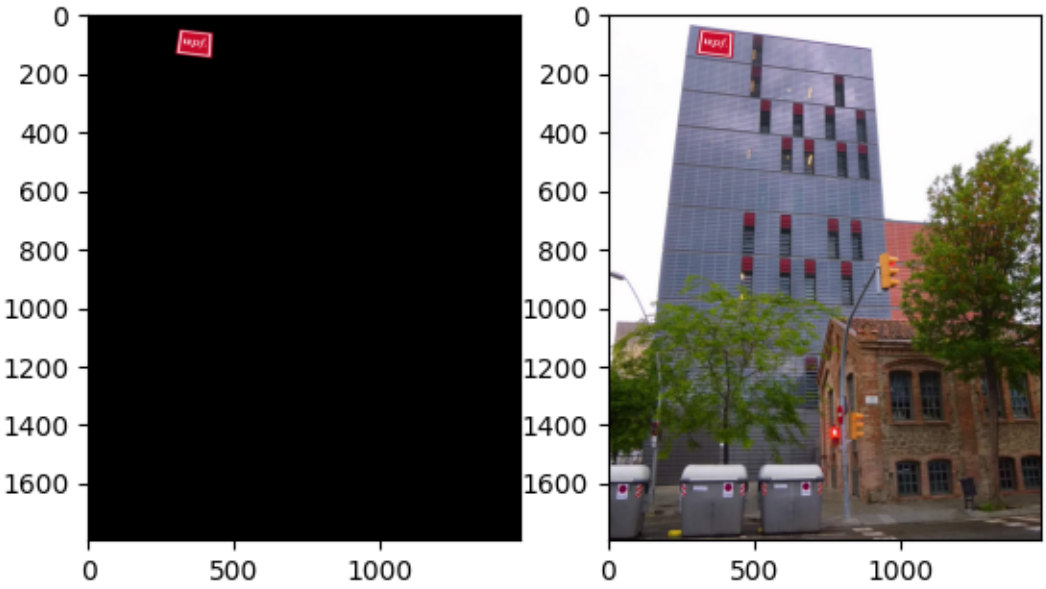


Figure 15: Result on warping the logo back to its original position with H^{-1}

Figure 16: Result on replacing the logo with several images.

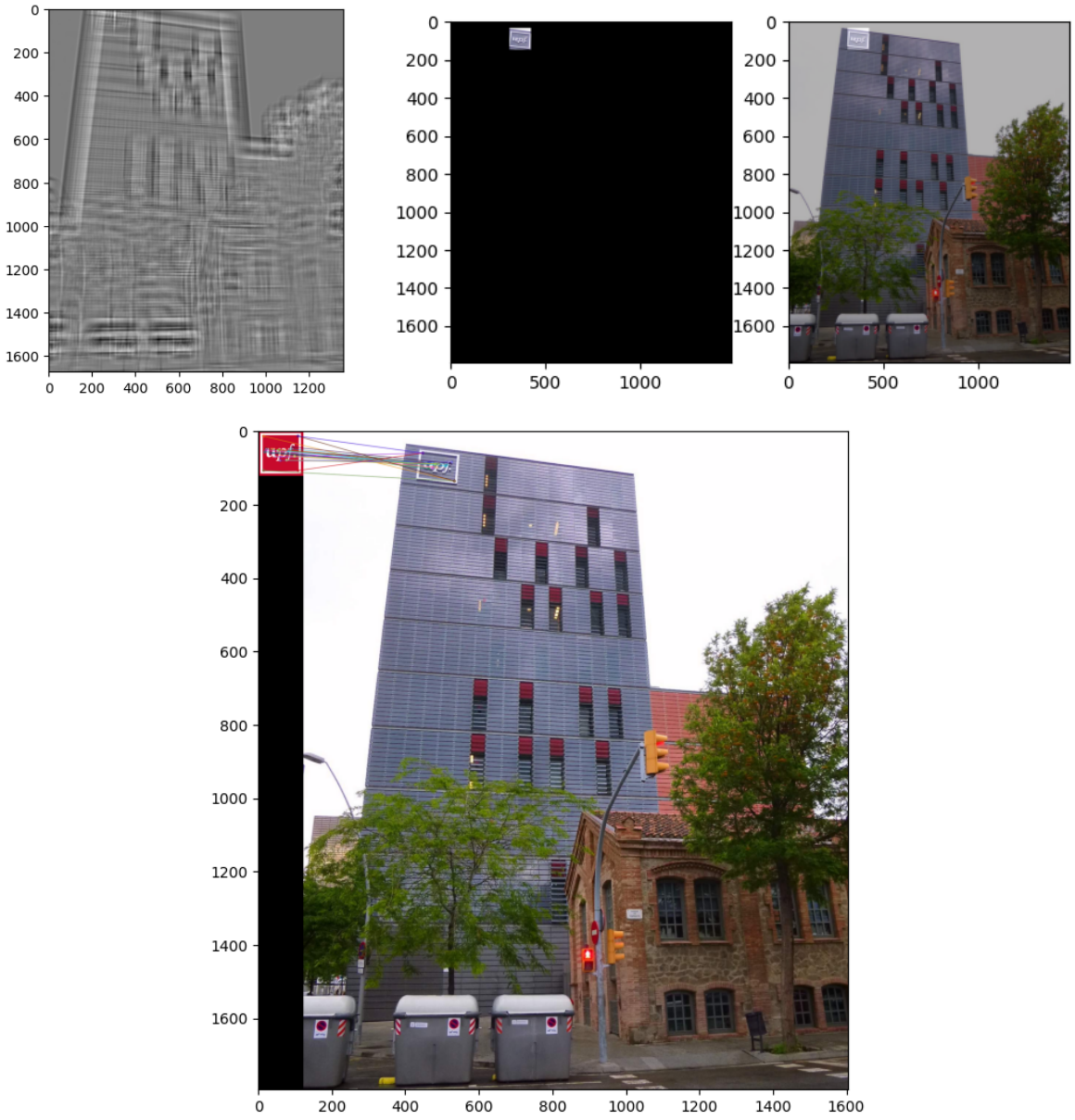


Figure 17: Result of using the convolution operation with the template as kernel (left) and region around the maximum signal point (right). Result of keypoint matching taking into account the detected RoI (bottom).

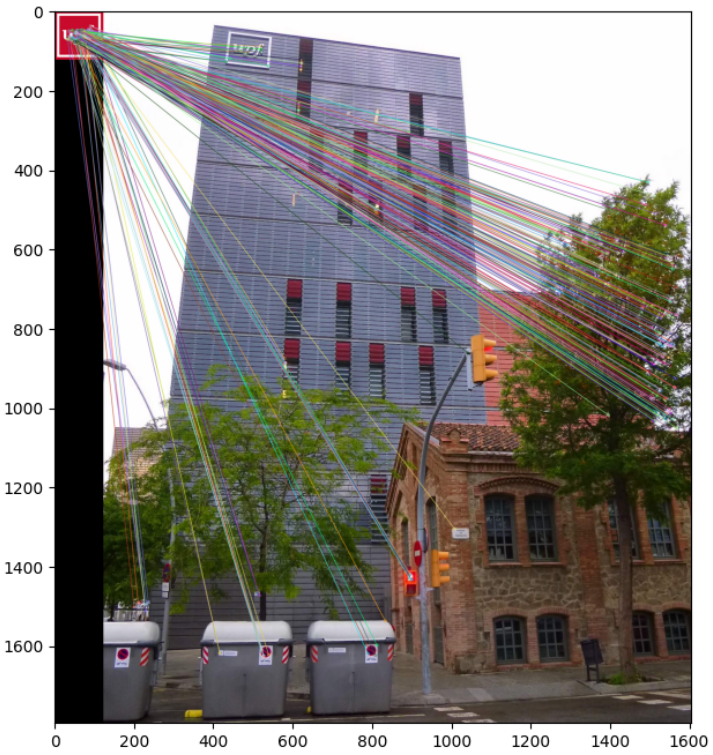


Figure 18: Due to the small number of relevant features, it's hard to establish robust correspondences between the template and the target.