Module 4 - Report week 2

Hicham El Muhandiz[1[0000−1111−2222−3333]] , Julia Ariadna Blanco Arnaus[1[0000−1111−2222−3333]] , Marcos Muñoz González[1[0000−1111−2222−3333]] , Rosana Valero Martínez[1[0000−1111−2222−3333]]

Universitat Autònoma de Barcelona, Campus UAB, Edifici O, 08193 Cerdanyola del Vallès, Barcelona

1 Introduction

In this project, we estimate a homography relating two images given a set of correspondences between them. In particular, we compute the homography using the Robust normalized DLT algorithm and then refine it using the Gold-Standard algorithm. The purpose of this project is to see different applications with these techniques, including creating image mosaics (panoramas), calibrating cameras with a planar pattern, implementing augmented reality and, detecting and inserting logos into an image.

2 Implementation

2.1 Homography estimation with the DLT algorithm

To do the tasks this week, we are assuming that we do not know the correspondences or relationship between two images (homography matrix). We will automatically estimate keypoints using an ORB detector and perform the homography matrix estimate with the DLT algorithm.

2.2 Compute image correspondences

The first step is to retrieve the keypoints from the images using ORB. This will return a list of keypoints for each image. To obtain the correspondences, keypoints are matched using a brute force matcher. Finally, the best matches are filtered by applying Lowe's ratio test. Given a pair of keypoints (correspondence), it will be kept if the distance between that keypoint and the second-best match is large enough, else it will be discarded due to ambiguity. Using this method, we can obtain a set of correspondences like the one shown in Fig 1.

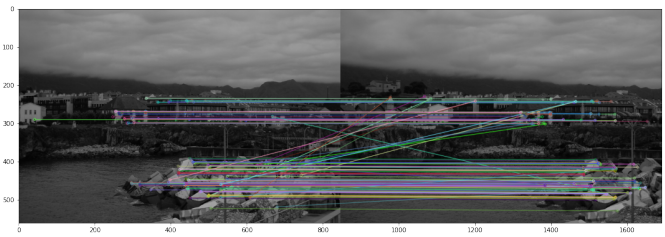


Fig. 1: Detected keypoints in images llanes_a.jpg and llanes_b.jpg and their correspondences (good matches)

2.3 Preliminaries to compute the homography (DLT algorithm)

This section describes the steps of the DLT algorithm implementation and auxiliary functions needed to estimate an homography between image pairs. First of all, before applying the DLT algorithm, we have to normalize the two sets points we want to estimate the homography matrix of. In particular, we want to move the centroid of each set of points to (0, 0) and set their average distance from the origin to $\sqrt{2}$. To do so, we must do the following:

Find scale and translation values On the one hand, the translation values to center the set of points will be the ones that cancel out the mean of each axis ($-mean\_points_x, -mean\_points_y$). On the other hand, the scaling value **s** can be obtained by dividing by the average distance and multiplying by $\sqrt{2}$ as it can be seen in Equation 1.

$$s = \frac{(\sqrt{2}n)}{\sum_{i=1}^{n} \sqrt{(x_i - mean\_points_x)^2 + (y_i - mean\_points_y)^2}} \tag{1}$$

Generate normalizing matrix T Once we found the amount of translation and scaling needed to normalize our points, we can transform them using a similarity transformation that has s and $-mean\_points_x, -mean\_points_y$ as scaling and translation factors respectively (Equation 2).

$$T = s \begin{pmatrix} 1 & 0 & -mean\_points_x \\ 0 & 1 & -mean\_points_y \\ 0 & 0 & 1/s \end{pmatrix} \tag{2}$$

After implementing this, we can generate the normalizing matrix for a pair of correspondences (p_i, p_i') and transform them so that we can apply the DLT algorithm.

Generate A_i and A The first step of the DLT algorithm is to compute the A_i matrix of each set of correspondences p_i, p_i'. This matrix is defined in Equation 3, where x_i, y_i and w_i are the projected coordinates of each point p_i (likewise for the points of the second image p_i').

$$A_i = \begin{pmatrix} 0^T & -w_i' p_i^T & y_i' p_i^T \\ w_i' p_i^T & 0^T & -x_i' p_i^T \end{pmatrix} \tag{3}$$

Finally, we can compute the matrix A by stacking all the A_i matrices of each correspondence (Equation 4).

$$A = \begin{pmatrix} A_1 \\ ... \\ A_n \end{pmatrix} \tag{4}$$

To obtain the homography estimate we obtain the SVD decomposition of A ($A = UDV^T$). We can construct H_{norm} (normalized homography matrix) as seen in Equation 5, where h is the last column of V.

$$H_{norm} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \quad h = \begin{pmatrix} h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9 \end{pmatrix}^T \tag{5}$$

With this we have already estimated H_{norm} and we only have to undo the normalization step to obtain the homography matrix H: $H = T'^{-1} H_{norm} T$.

Inliers An auxiliary function needed to compute the homography is to return the inliers of each set of points. To do so, the putative distance is computed for each pair of points (Equation 6, where $\hat{x}$ is the estimate of the x coordinate using x' and H and vice versa). Inliers will be the points where the putative distance is under a threshold th.

$$putative\_distance = \sqrt{(\hat{x} - x)^2 + (\hat{y} - y)^2 + (\hat{x'} - x')^2 + (\hat{y'} - y')^2} \tag{6}$$

2.4 Compute the homography (DLT algorithm) between image pairs

To compute the homography, we proceed using the DLT algorithm with RANSAC selecting a set of correspondences (that we have computed in the previous step, explained in the previous section) in each iteration. We determine the list of indices corresponding to inlier matches. Using this, we select the homography with the most inliers as the final outcome. We obtain then the homography between two images, as we can see in Fig 2.

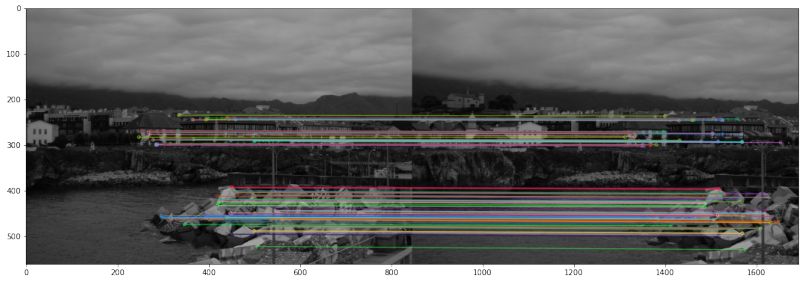


Fig. 2: Homography (DLT) of the images llanes_a.jpg and llanes_b.jpg

2.5 Application: Image mosaics

One of the applications of the homographies is that we can create a mosaic by properly transforming and fusing the different image contents.

To do so, we consider three images for each image set and we compute the homography matrix between images 1 and 2 (H_12) and the homography matrix between images 2 and 3 (H_23). Once we have estimated the homographies relating the images, we have to apply some transformations to each of the three input images. The first image is transformed using H_12, the second image is left unchanged using the identity matrix, and the third image is transformed using the inverse of H_23.

2.6 Refinement of the estimated homography with the Gold Standard algorithm

In this step, we try to minimize the error of our estimated homography transformation obtained from the DLT algorithm that relates the point inliers p and p' from the first and second image respectively. To do this, we use the non-linear least squares method with the Levenberg-Marquardt algorithm. We first compute an initial estimate of the set of points p given by the homography transformation, $\hat{p}$. Then, we define the cost function to be minimized for the least squares algorithm. We use the Symmetric Transfer Error Equation 7 to take into account both the points and their transformations given by the homography of the first and second image

$$\sum_i d(p_i, \hat{p}_i)^2 + d(p'_i, \hat{p}'_i)^2 \tag{7}$$

where p_i and p'_i are the point inliers of image 1 and 2 respectively and $\hat{p}_i$ and $\hat{p}'_i$ are their given transformations by the homography.
It must be noted, that to compute the Symmetric Transfer Error we need the points in non-homogenous coordinates. Finally, we compute the least squares algorithm using our defined cost function to obtain the refined points p_refined and homography_refined transformation matrix.

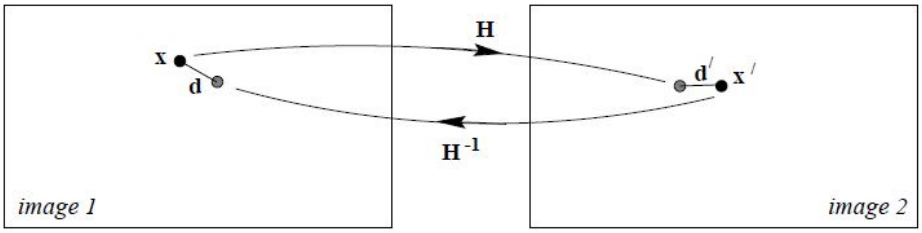


Fig. 3: An illustration of the symmetric transfer error from [1]

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

Another application of the homographies is the camera calibration with a planar pattern by using Zhang's algorithm. This algorithm calibrates a camera using N views of the planar pattern. A camera can be modeled as the relationship between a 3D point X and its image projection, as x = PX, where P is the camera matrix. We can decompose $P = K[R|t]$, being K the camera calibration matrix which contains the internal orientation of P, and R and t the external parameters of the camera pose (position and orientation of the camera in the reference system of the world).
It is based on two conditions: the pattern plane should be z = 0 to allow us to rewrite $[h1h2h3] = [Kr1Kr2Kt]$, and r1 and r2 should be columns of an orthogonal matrix. From this, we define the image of the absolute conic, as $\omega = K^{-t}K^{-1}$, which is a matrix 3x3:

$$\omega = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{12} & w_{22} & w_{23} \\ w_{13} & w_{23} & w_{33} \end{pmatrix} \tag{8}$$

and Ω as:

$$\Omega = \begin{pmatrix} w_{11}, w_{12}, w_{13}, w_{22}, w_{23}, w_{33} \end{pmatrix}^T \tag{9}$$

The algorithm starts estimating the n homographies that relate the planar pattern and the n images, and then builds the matrix V based on the following expression:

$$h_i^T \omega h_j^T = \begin{pmatrix} h_{1_i} h_{2_i}, h_{3_i} \end{pmatrix} \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{12} & w_{22} & w_{23} \\ w_{13} & w_{23} & w_{33} \end{pmatrix} \begin{pmatrix} h_{1_j} \\ h_{2_j} \\ h_{3_j} \end{pmatrix} = V_{ij}^T \Omega \tag{10}$$

where V_{ij}^T:

$$\begin{pmatrix} h_{1i}h_{1j}, h_{1i}h_{2j} + h_{2i}h_{1j}, h_{1i}h_{3j} + h_{3i}h_{1j}, h_{2i}h_{2j}, h_{2i}h_{3j} + h_{3i}h_{2j}, h_{3i}h_{3j} \end{pmatrix} \tag{11}$$

This allows us to obtain the following two equations:

$$\begin{cases} V_{12}^T \Omega = 0 \\ (V_{11}^T - V_{22}^T)\Omega = 0 \end{cases} \tag{12}$$

As we have 2 equations for 6 unknowns we need at least 3 views (homographies) to find ω, and we have to solve $V_{2nx6}\Omega = 0$.

In practice, Ω is the singular value associated to the smallest singular value and can be found by applying SVD to the matrix V we have created and picking the last column of $\bar{U}$ ($V = UD\bar{U}^T$)

Finally, K can be found by applying Cholesky to ω (rearrange Ω) and computing the inverse, as $\omega = K^{-T}K^{-1}$.

3 Results

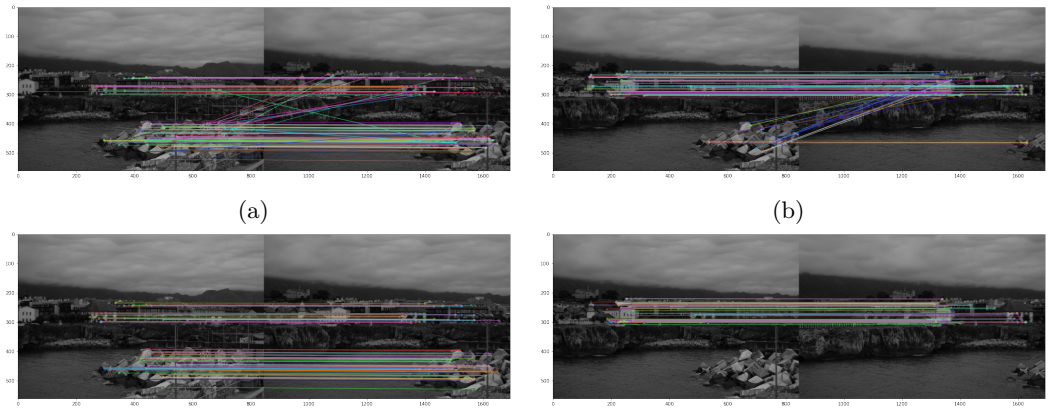
3.1 Application: Image mosaics

In our case, we have four different image sets with three images for each one.

As we have explained in the previous section, to build the mosaic, we have computed the correspondences and the homographies for the four datasets, as we show in Fig 4, Fig 5, Fig 6 and Fig 7.

We can observe in Fig 8(a) and Fig 8(c), where we have built the mosaic for the image set of *Llanes* and *Aerial site 13*, respectively, that our results are pretty good in both cases. This is as expected because, as we can see in Fig 4 and Fig 6, we have calculated a good estimation of the homographies.

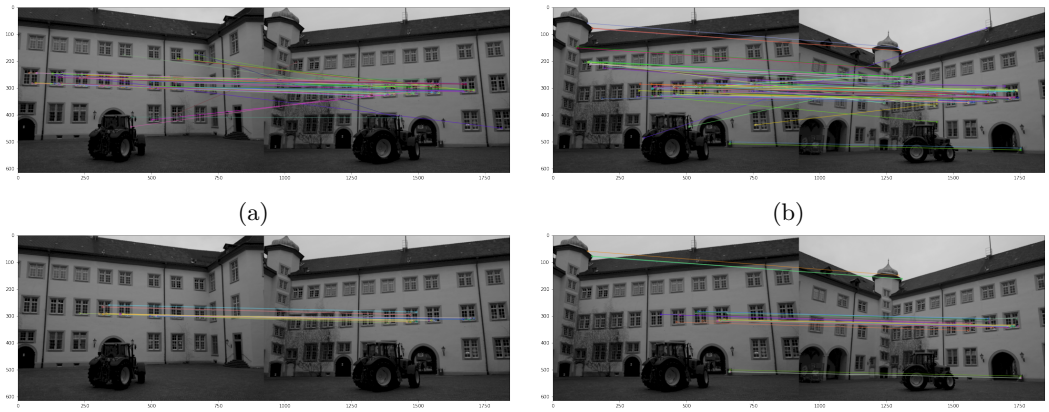
However, in Fig 8(b) and Fig 8(d), *Castle* and *Aerial site 22* image sets, respectively, we have the opposite case. We haven't obtained good results of the correspondences and homographies. This is because the keypoints are very similar throughout the image and they can not be well differentiated, as shown in Fig 5 and Fig 7.



(a) (b)

(c) (d)

Fig. 4: Image set *Llanes*. a) and b) correspond to the detected keypoints and their correspondences in images llanes_a.jpg and llanes_b.jpg, and in images llanes_b.jpg and llanes_c.jpg, respectively. c) and d) correspond to the homography (DLT) of the images llanes_a.jpg and llanes_b.jpg, and images llanes_b.jpg and llanes_c.jpg, respectively



(a) (b)

(c) (d)

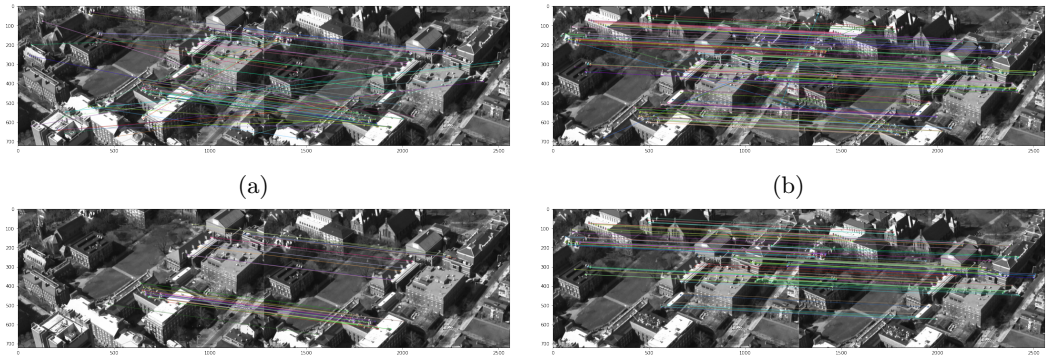
Fig. 5: Image set *Castle*. a) and b) correspond to the detected keypoints and their correspondences in images castle_0014_s.jpg and castle_0015_s.jpg, and in images castle_0015_s.jpg and castle_0016_s.jpg, respectively. c) and d) correspond to the homography (DLT) of the images castle_0014_s.jpg and castle_0015_s.jpg, and images castle_0015_s.jpg and castle_0016_s.jpg, respectively

3.2 Refinement of the estimated homography with the Gold Standard algorithm

There are two important results that we will show by using the Gold Standard algorithm:

- The contrast between the least squares error of the refined and original points, and the resulting image mosaics.
- The different results are given by the Gold Standard algorithm as the effect of using the RANSAC algorithm in DLT.

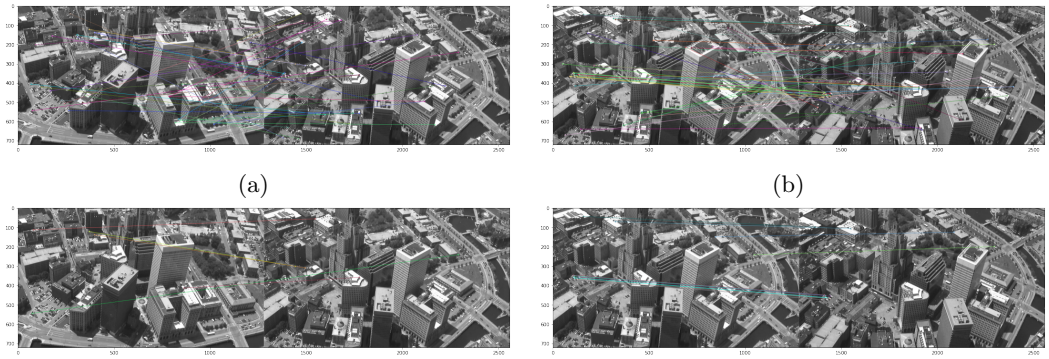
First, we show the results of the refined homography transformation given by the Gold Standard algorithm in Fig 9. The related initial error on the first image of Fig 9 given by the Symmetric Transfer Error function in the least squares algorithm is 591283872.5520 whether as the final square error after performing the algorithm is 25.5236. Similarly, on the second image of Fig 10, the error is 305574578.1093 whether and the final square error is 36.9892. By visual inspection, it can be seen that no big differences are found on both image mosaics in Fig 10, but the difference in initial and final Symmetric Transfer Error is substantial by using the least-squares algorithm.



(a) (b)

(c) (d)

Fig. 6: *Aerial site 13*. a) and b) correspond to the detected keypoints and their correspondences in images frame00000.png and frame00002.png, and in images frame00002.png and frame00003.png, respectively. c) and d) correspond to the homography (DLT) of the images frame00000.png and frame00002.png, and images frame00002.png and frame00003.png, respectively.



(a) (b)

(c) (d)

Fig. 7: Image set *Aerial site 22*. a) and b) correspond to the detected keypoints and their correspondences in images frame_00001.tif and frame_00018.tif, and in images frame_00018.tif and frame_00030.tif, respectively. c) and d) correspond to the homography (DLT) of the images frame_00001.tif and frame_00018.tif, and images frame_00018.tif and frame_00030.tif, respectively.

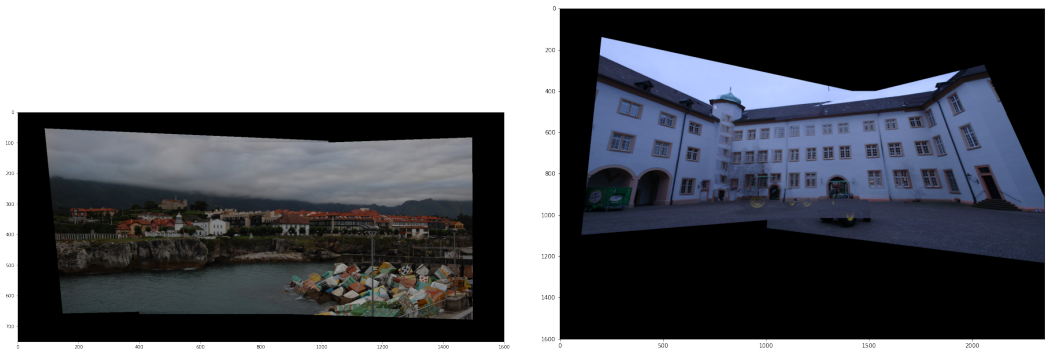
Second, we show that the Gold Standard algorithm can give very different results given the same implementation on several iterations. That is because that RANSAC (an algorithm that uses randomness) is used in the DLT algorithm to increase the robustness to outliers and then the resulting points and the homography transformation matrix of the DLT algorithm are given to the Gold Standard algorithm. Observe in Fig 17 that both mosaics are really different and visually we can see that one mosaic is more or less appealing visually whether the other one is not.

3.3 Camera calibration

By performing camera calibration using Zhang's algorithm, we obtained a matrix K, containing the internal parameters of the camera, which is defined as:

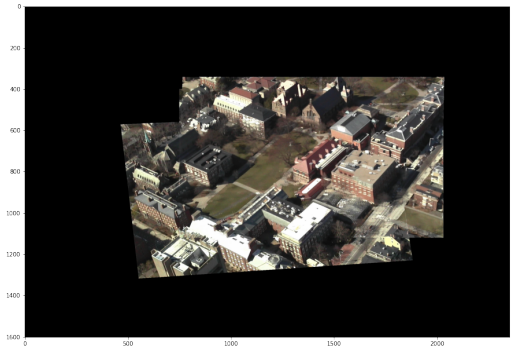
$$K = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{13}$$

where f_x and f_y are the focal lengths on the x and y axis, p_x and p_y are the coordinates of the principal point and s is the skew factor. In our case, we have $f_x = 2.792 \times 10^3$ and $f_y = 2.779 \times 10^3$. Both values differ due to errors in the calibration process and distortion. The skew parameter $s = -1.009$ tells us that we have a radial distortion. Finally, the principal points obtained are

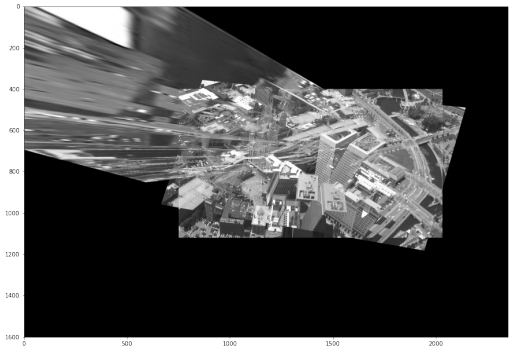


(a) Image set *Llanes*

(b) Image set *Castle*



(c) Image set *Aerial site 13*



(d) Image set *Aerial site 22*

Fig. 8: Mosaic images obtained for each image set

$f_x = 9.104 \times 10^2$ and $f_y = 7.08 \times 10^2$. To compute Cholesky's decomposition, we had to use the Scipy package as the Numpy package computes the lower triangular matrix, while K is the upper triangular matrix. Once we have the internal parameters we can retrieve the external parameters:

$$\begin{cases} r_1 = \frac{K^{-1}h_1}{||K^{-1}h_1||} \\ r_2 = \frac{K^{-1}h_2}{||K^{-1}h_2||} \\ t = \frac{K^{-1}h_3}{||K^{-1}h_1||} \end{cases} \tag{14}$$

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

The camera positions can be represented with respect to the plane, as in the first image of Figure 11, or considering the camera is fixed and the image plane is moving, as in the second image of the same figure.

We have also tried to use a different number of views to compute the camera's internal parameters. The results obtained using 5 views are shown in Figure 12. By using more views of the planar pattern, the calibration errors and the distortion in the matrix parameters K should be reduced.

3.4 Augmented reality

To insert a virtual object (in this case a 3D cube) in the image, we define the corners of a 3D cube and use the projection matrix from the estimated homographies to project the corners onto the image plane. To ensure a proper representation of the inserted object, we use 20 corners from the five planes of the cube and add an offset to center the cube at the planar pattern. In order to augment the real world, the real and virtual objects must be accurately positioned relative to each other, so the intrinsic camera parameters are used to project 3D points onto the image plane to

Fig. 9: Two images with their points refined by the Gold Standard algorithm. Original points are + in cyan whether as refined points are + in fuchsia

Fig. 10: On top the original mosaic given by the DLT algorithm homography. On bottom the mosaic computed by using a refined homography from the Gold Standard algorithm. In this case, no big visual difference is found.

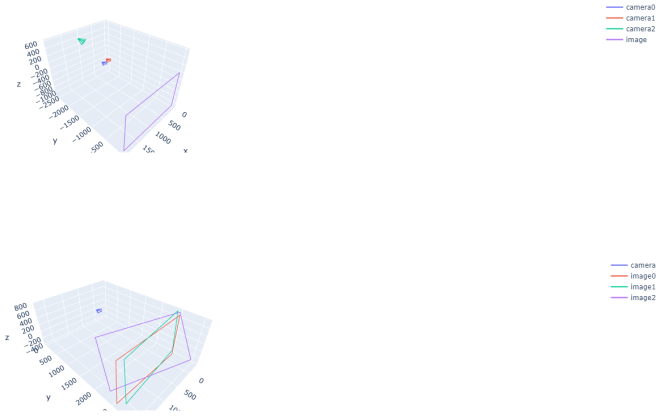


Fig. 11: Planes positions and camera position using 3 views in Zhang's algorithm

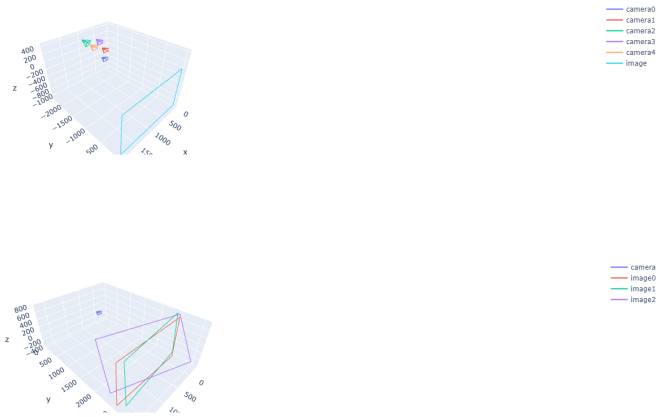


Fig. 12: Planes positions and camera position using 5 views in Zhang's algorithm

match the pose and optical properties of the real and virtual camera. In Figure 13 we can see how the virtual object is properly inserted:

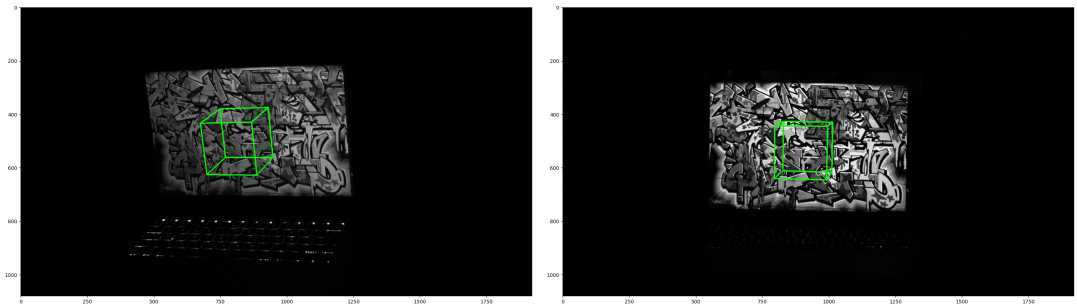


Fig. 13: 3D Virtual object inserted on top of the flat pattern

3.5 Logo detection

We have performed logo detection using the DLT algorithm. We compute the points with two feature detectors, ORB and SIFT. With ORB in Fig 14 we don't have enough keypoints to relate the logo with the target image, so the logo is not correctly detected and some other part of the image is detected as a logo instead. By contrast, with SIFT we have enough features on the logo target image so the logo can be correctly detected as can be seen in Fig 15.

Fig. 14: Logo detection using ORB. On the left the found corresponds with the DLT algorithm and on the right on green the detected wrongly logo

3.6 Logo replacement

For logo replacement, we have used the DLT algorithm. First, we compute the logo detection using SIFT since it was the best-performing feature detector and then we replace the logo on the original image using the obtained homography transformation with the DLT algorithm. As a particular note, if the logo detection part does not work correctly the logo replacement will not work. On the shown images in Fig 16 it can be seen that our logo replacement algorithm works well.

Fig. 15: Logo detection using SIFT. On the left the found corresponds with the DLT algorithm and on the right, on green the correctly detected logo

Fig. 16: Logo replacement using SIFT

Fig. 17: On top the original mosaic given by the DLT algorithm homography. On bottom the mosaic computed by using a refined homography from the Gold Standard algorithm. Observe the visual appeal difference between of both results.

4 Conclusions

In this work, we have implemented and evaluated several techniques for homography estimation and applications. The DLT algorithm and its variants have been tested and some insight into the implementation and results of the algorithms are shown.

Some problems that appeared during the implementation of the exposed techniques are:

- The least squares error function of the library Scipy needs the input variables size to be the same as the error function output size. Even if the reasoning for how to use the least squares function was correct, the function had some constraints on how we should it.
- To compute Cholesky's decomposition it's required to the matrix V to be positive definite. Sometimes, the homographies obtained by the Ransac-DLT algorithm caused the matrix V to not meet this requirement, and therefore we couldn't obtain Ω. We had to rerun the homography estimation part to obtain a matrix V positive definite.
- If there are not enough meaningful keypoints to relate two images there may be unexpected behavior. For that reason, exceptions such as this should be handled in some cases such as logo detection or replacement.

Overall this has been an interesting project, where we have seen both the inner implementations for homography estimation and their results. As a relevant bibliography, we have used the book Multiple View Geometry in Computer Vision by Richard Hartley and Andrew Zisserman [1] and the slides provided by Raul Queiroz Feitosa [2].

References

1. Hartley, R., Zisserman, A. (2004). Multiple View Geometry in Computer Vision (2nd ed.). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511811685
2. Homography estimation slides, https://web.archive.org/web/20150929063658/http://www.ele.puc-rio.br/ visao/Topicos/Homographies.pdf. Last accesed 17 Jan