Module 4 - Report week 1

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 implemented a system to perform homographies on a given image and correct them. In particular, the given images present warped lines due to their perspective, which are undone by performing affine and metric rectifications. The purpose of this project is to be able to restore the angles and length ratios. The main limitation of the method is that a set of parallel and orthogonal lines have to be specified beforehand, so it cannot be automated.

2 Implementation

2.1 Planar transformations

There exist several types of planar transformations such as similarity, projective or affine transformations. Each one has several properties and transforms the image plane in different ways.

Similarity transformation Similarity transformations require a scaling parameter, a rotation one, and two translation ones (x and y directions respectively). The scaling resultant of this transformation is isotropic, meaning that with this transformation we can rotate, scale or move the image without affecting the length ratios or the parallelism in it. The implementation consisted of using the following equation 1 as the homography matrix and leaving the scale, angle, and translations as user-inputted parameters.

$$x' = H_s x = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} x \tag{1}$$

Two examples can be seen below.

Affine transformation In the case of affine transformations, several properties are preserved like parallelism or the ratio of lengths on parallel lines. A matrix to perform this type of transformation is shown in equation 2 where we have 6 degrees of freedom (2 for scaling, 2 for the rotation angles, and 2 for translation).

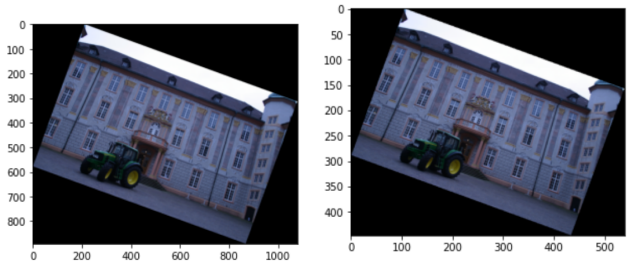

Fig. 1: Rotation by 20 degrees. 0.5 scaling and rotation by 20 degrees

Following this, we can decompose the affinity matrix transformation into four different transformations by using singular value decomposition given the case that our matrix is non-singular. The affine transformation then is decomposed into two rotations, one scaling, and one translation. This decomposition is one effective way of ensuring that our transformation is an affine one since we can see if the matrix produced by the singular value decomposition is equal to the initial one. Also, we can verify that by using the four transformations sequentially we get the same result as using just the affine transformation on an image.

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_x \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Fig. 2: An affine transformation by sheering 0.5 and a translation of 1 on the x-axis

Projective transformation The implementation of the projective homography matrix is the same as with the affine transformation, directly defining each parameter (3). However, this time we have 2 more degrees of freedom (v1 and v2) that allow operations that do not preserve parallelism (changes in perspective). In particular, the 2 new parameters are very sensitive to small changes as

can be seen in Fig 3.

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{32} & h_{32} & h_{33} \end{bmatrix} \tag{3}$$

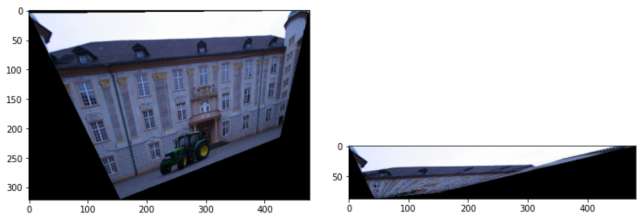

Fig. 3: v1, v2 = (0.001,0 0.0015). (v1, v2 = 0.001, 0.01)
Both examples use A = [[1,.5], [0,1]] and no translation

2.2 Preliminaries for affine and metric rectification

Before performing affine or metric rectification techniques we have to obtain the representation of different interactions between lines and points.

Computation of vanishing line To compute the vanishing line from two points we simply compute the cross-product between both points. This can be done by using the equation .

$$\vec{a} \times \vec{b} = |a||b| \sin\theta n \tag{4}$$

Computation of vanishing points In order to compute the vanishing points, we can also utilize the cross product, in this case, each vanishing point will be obtained through two different lines by using equation 4.

Transformation of lines We obtain the diagonal lines between 4 sets of parallel lines by using again the cross-product with equation 4. We first compute the corner points of each pair of lines with the cross product and then we compute a pair of diagonal lines by using the previously obtained corner points.

Getting angles between lines Angles have been obtained using the dual conic (5) to obtain the cosine and then compute the angle.

$$\cos\theta = \frac{l^T C_\infty^* m}{\sqrt{(l^T C_\infty^* l)(l^T C_\infty^* m)}} \tag{5}$$

2.3 Affine rectification

To perform affine rectification we have to choose two sets of parallel lines so that we can find their vanishing points and revert them to restore their parallelism in the rectified image. In Fig 4 the lines can be seen. In this step, however, the diagonal lines still will not be used.

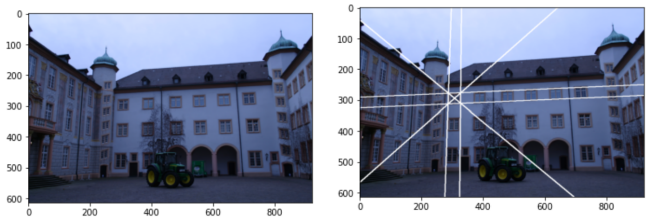

Fig. 4: Original image and lines

2.4 Metric rectification

To perform stratified metric rectification we need to have the affine rectification homography matrix beforehand, and the affine transformation of 2 pairs of orthogonal lines. With this, we can build an equation (7) to solve for the s matrix (6) with each pair of transformed orthogonal lines finding the null space.

$$s = (s_{11}, s_{12}, s_{22})^T \tag{6}$$

$$(l'_1 m'_1, l'_1 m'_2 + l'_2 m'_1, l'_2 m'_2)s = 0 \tag{7}$$

Afterwards, we obtain K using the Cholesky decomposition of S and invert it. Combining this matrix with the one obtained in the affine rectification step, we can compute the transformation matrix and apply it to our image.

3 Results

3.1 Affine rectification

The results for the affine rectification can be seen in Fig 5 and Fig 6.

To evaluate the results, we compute the angle between the different pair of lines before and after the affine rectification. As we can see on Tables 1 and 2, we can show that parallelism is preserved because both angles have been set to either 0 or 180 (180 because we may have defined the line in the opposite direction).

As can be seen in both examples, the parallelism of the lines has been restored, although it is not quite noticeable in the first image given that the original one was already not as warped by the perspective.

Table 1: Angles between the different pair of lines before and after the affine rectification on image 0000_s.png

	Angle between $\mathbf{L}_1$ and $\mathbf{L}_2$ (^{o})	**Angle between $\mathbf{L}_3$ and $\mathbf{L}_4$ (^{o})**
Before	0.099182	178.656463
After	0.000001	180.000000

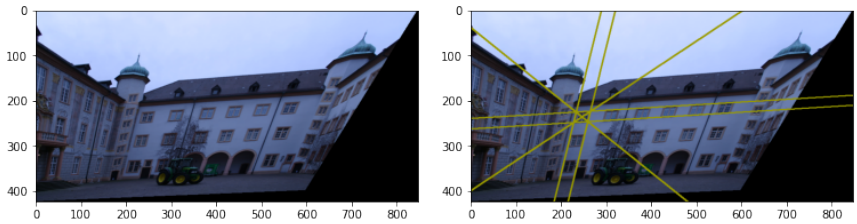

Fig. 5: Affine rectification of image 0000_s.png

Table 2: Angles between the different pair of lines before and after the affine rectification on image 0001_s.png

	Angle between $\mathbf{L}_1$ and $\mathbf{L}_2$ (^{o})	**Angle between $\mathbf{L}_3$ and $\mathbf{L}_4$ (^{o})**
Before	4.420805	0.124941
After	0.000000	0.000001

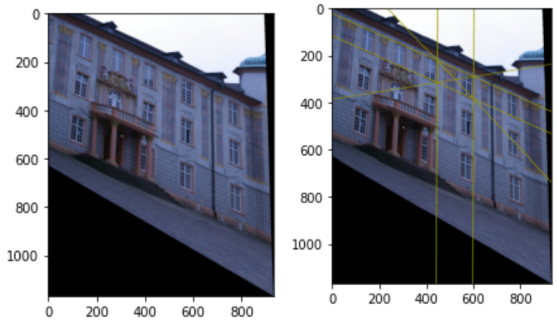

Fig. 6: Affine rectification of image 0001_s.png

3.2 Metric rectification

The results of metric rectification can be seen in Fig 7 and Fig 8. Qualitatively, we can see how the lines forming 4 side polygons have been transformed into squares or rectangles.

We can observe in Table 3 that the angles between the different pairs of lines after the metric rectification. As seen in the bottom field, both angles of the windows have been restored to 90 degrees after the affine and metric transformations, meaning that our goal to compute the original angles of the building was achieved as windows are most likely to be straight and have perpendicular edges.

Table 3: Angles between the different pair of lines before and after the metric rectification on image 0000_s.png

	Angle between $\mathbf{L}_1$ and $\mathbf{L}_2$ (^{o})	**Angle between $\mathbf{L}_3$ and $\mathbf{L}_4$ (^{o})**
Before	85.454019	93.103262
After	90	90

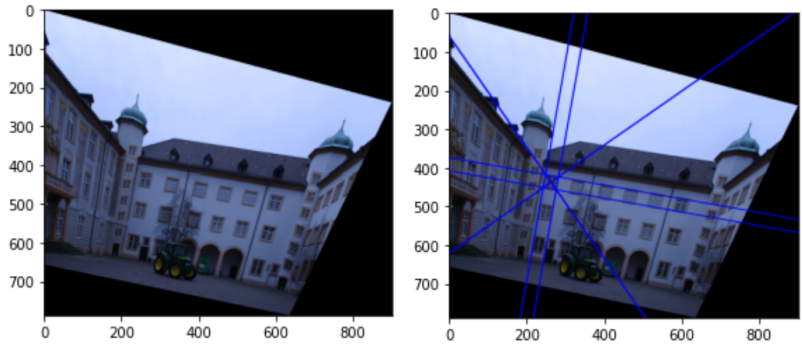

Fig. 7: Metric rectification of image 0000_s.png

Table 4: Angles between the different pair of lines before and after the metric rectification on image 0001_s.png

	Angle between $\mathbf{L}_1$ and $\mathbf{L}_2$ (^{o})	**Angle between $\mathbf{L}_3$ and $\mathbf{L}_4$ (^{o})**
Before	67.857261	72.153125
After	90	90

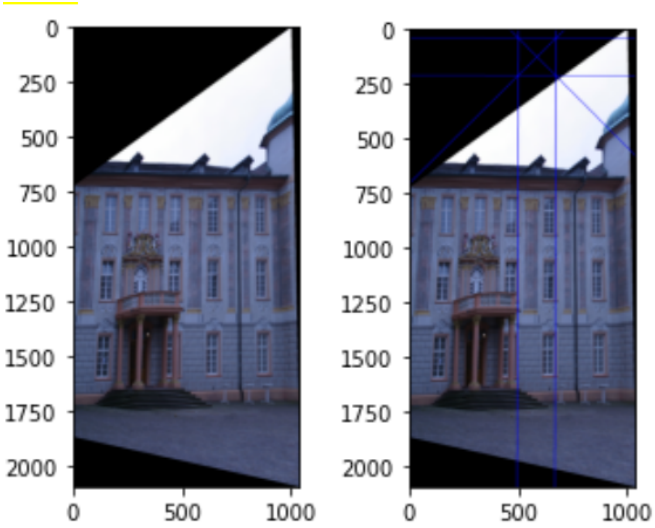

Fig. 8: Metric rectification of image 0001_s.png

4 Conclusions

In this work, we have implemented and evaluated several 3D vision techniques in the context of urban scenes. In particular, we used different types of image transformations and learned about how to perform affine and metric rectification on images.

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

- Cholesky decomposition only accepts positive definite matrices: Even though this was not a problem initially because we could compute the metric rectification of some images with it, we noticed that on a specific image, it did not work. It turned out that we were incorrectly computing the lines of that image and so it prevented us from looking for another error.
- When estimating the angles with the dual conic, in one of the cases, the value outputted was slightly out of bounds for the arc cosine (-1.0000000000000002), so an assignment to ensure that the values were within bounds had to be added. We attribute this issue to an estimation error of this formula.
- Lack of information on some indispensable tools: For instance, even though Scipy has good documentation, there were not many examples of applications of some functions such as scipy.ndimage.map_coordinates.

Overall this has been an interesting project, even though because of the lack of real examples it may have been more difficult than initially thought. As a relevant bibliography, the book Multiple View Geometry in Computer Vision by Richard Hartley and Andrew Zisserman [1] has been an indispensable resource in order to perform some steps of the project.

References

1. Hartley, R., Zisserman, A. (2004). Multiple View Geometry in Computer Vision (2nd ed.). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511811685