M6 3D RECOVERY OF URBAN SCENES - SESSION 1

AXEL BARROSO & SERGIO CASTRO & SERGIO SANCHO

GOAL

The goal of this session is to learn the basic concepts and techniques of homography, play with different planar transformations and apply metric and affine transformations to given images.

1 INTRODUCTION

Apply homography to images is a well known topic in computer vision. We will apply some kind of liner transformation to one image, and then we will get another different image. Depending on the objective we are looking for, the input image will suffer one transformation or another. In the very beginning, we define the matrix that represents that transformation, and just by multiplying image points and matrix, we come out with our new image.

Multiplying matrix transformation and image points is not as easy as it seems. There are many different techniques and approaches for applying matrix transformations to images. In our project, we defined two different techniques. The first technique is to keep image sizes and coordinates fixed, we then can lose part of the resultant image (depends on the transformation applied) or fitting the resulting image in the needed size (based on the 4 corners of the image). In our case, we have mostly always decided to fit the image in the original size based on its 4 corners in order to not lose any information, being able to see always the effect of the transformations applied. Although, we also use the other approach in order to see the effect of planar transformations such as translations.

A bit more into detail, we start with an image and a matrix. We come up with the idea of building a mesh of points in order to represents every possible point coordinate. We then apply our matrix H to our group of points to know their new coordinates by using the interpolation function provided by Matlab.

2 IMAGE TRANSFORMATIONS

2.1 Similarities

We were asked to play around with affine transformation. As first step, we explain the shape of similarity matrices. Afterwards, we will show series of images after specific transformations such as translations, rotations, scales. A similarity transformation is an isometry composed with an isotropic scaling. This transformations performed are part from similarities transformations, which follow the next form:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Figure 1: Similarity matrix representation

Thus, we can see some transformations that we applied to the original image:

Figure 2: Original image

Figure 3: Translation of vx=100 and vy=150

Figure 4: Rotation of 20º

Figure 5: Scale of 0.5 of the original image

Figure 6: Translation of vx=100 and vy=150 and rotation of 20°

2.2 Affinities

An affine transformation is a non-singular linear transformation followed by a translation. As similarity transformation, it has the matrix representation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Figure 7: Affine matrix representation

or in block form

$$\mathbf{x}' = \mathbf{H}_A \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

Figure 8: Affine matrix representation in block

with A as a 2x2 non-singular matrix. A planar affine transformation has six degrees of freedom corresponding to the six matrix elements. The transformation can be computed from the three point correspondences. Then, it is defined by 6 parameters:

- two angles
- two translation vectors
- two scaling factors

In our case, we have applied the following matrix:

$$\begin{vmatrix} 1.15 & -0.3 & 160 \\ 0.3 & 0.65 & -65 \\ 0 & 0 & 1 \end{vmatrix} \tag{1}$$

That gave us the result that can be seen in figure 9

Figure 9: Affine Transformation

Then, using the decomposition of our affine matrix we decomposed it in 4 different parts with the *Singular Value Decomposition*:

- two rotations
- one scale
- one translation

Afterwards, we verified that the correct product of these 4 decompositions gave the same result than the original matrix following the next expression:

$$A = R_\theta \cdot R_{-\phi} \cdot D \cdot R_\phi; H_1 = \begin{bmatrix} A & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

Figure 10: H decomposed in rotations, scale and translation

We can built these matrices using the *Singular Value Decomposition* as mentioned before. The expression can be seen in figure 11 where we can compute U, D, V.

$$[U, D, V] = svd(A)$$

Figure 11: Singular Value Decomposition

By using these matrices, we will be able to reach the two rotations and the scale shown in figure 10, whose form is shown in figure12

$$R_\theta = \begin{bmatrix} U \cdot V^T & 0 & 0 \\ 0 & & 0 \\ & & 1 \end{bmatrix}; R_\phi = \begin{bmatrix} V^T & 0 & 0 \\ 0 & & 0 \\ & & 1 \end{bmatrix}; S = \begin{bmatrix} \boldsymbol{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \\ & & 1 \end{bmatrix}; T = \begin{bmatrix} 1 & 0 & h_{13} \\ 0 & 1 & h_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 12: Expression of the Rotations, Scale and Translation

Result that can be seen in the figure 13

Finally, we verified that the sequence of the four previous transformations computed before produced the same original image, applying each transformation independently from the others. The result is shown in figure 14

Figure 13: Result Image after the Affine Matrix decomposition

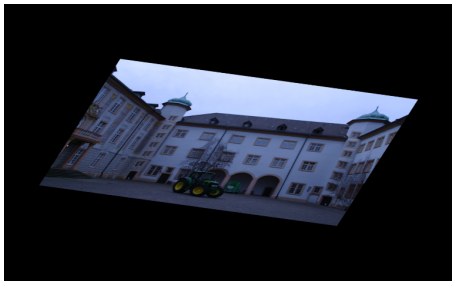

Figure 14: Result varying the sequence of the transformations

2.3 Projective Transformation

A projective transformation is a general non-singular linear transformation of *homogeneous* coordinates. This generalizes an affine transformation, which is the composition of a general non-singular linear transformation of *inhomogeneous* coordinates and a translation. Transformation keeps invariant the properties of:

- Concurrency
- Collinearity
- Order of contact
- Tangency discontinuities
- Cross ratio (ratio of lengths)

It has the matrix representation:

$$\mathbf{x}' = \mathbf{H}_{\mathrm{p}}\mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^{\mathsf{T}} & v \end{bmatrix} \mathbf{x}$$

Figure 15: Projective matrix representation

We have applied a Projective Transformation with the following matrix:

$$\begin{vmatrix} 0.7 & -1 & -0.5 \\ 0.2 & 0.8 & -1 \\ 0.00004 & 0 & -1 \end{vmatrix} \tag{2}$$

The result obtained can be seen in figure16

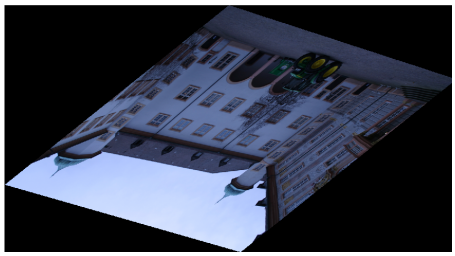

Figure 16: Result of the image after a Projective Transformation

3 AFFINE RECTIFICATION

The goal of the Affine Rectification is to delete the projective distortions of the image. In this part of the practicum, we will apply an affine rectification to the image from the figure 2.

In order to compute the transformation, we have the following information of the image points:

Figure 17: Information of the image

With this information, we will be able to compute the lines that are parallel, then compute their intersection point at the infinite (*Vanishing Point*) and finally compute the Vanishing Line that will tell us which transformation does the image need in order to erase the perspective view and in order to have real parallel lines from our view.

In our case, we have used the information of the points *424, 240, 712, 565* from the information image 17. With this points we have computed the lines that passes through them as:

$$\begin{aligned} ax + by + c &= 0 \\ line &= (a, b, c) \end{aligned} \tag{3}$$

The lines used can be seen in the figure 18

In order to compute the *Vanishing Points*, we used a cross-product between the lines. Once we had the two vanishing points computed, we used another cross product in order to compute the *Vanishing Line*. Note that in order to have the expression in *Cartesian*, it was necessary to divide the coordinates of the points by the third component.

Figure 18: Lines used

After computing the Vanishing Line, we could create our Affine Matrix with the following form:

$$\ell = (l_1, l_2, l_3)$$

$$\mathbf{H}_{a\leftarrow p} = H_a \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{pmatrix}$$

Figure 19: Formula for Affine Matrix

The results we obtained are shown in the figure 20

Figure 20: Result after Affine Transformation

The angles computed between the lines are shown in the table 1

Angles º	**Upper Left**	**Upper Right**	**Lower Left**	**Lower Rigth**
Previous	85.5	86.8	85.6	86.9
Transformed	72.7	72.6	72.7	72.6

Table 1: Angles before and after the Affine transformation

4 METRIC RECTIFICATION

The objective of the Metric Rectification is to recover the metrics properties lost in the Affine Rectification. The main information that it uses is the angle between two lines.

In figure 21 the lines that we have used in order to compute the Metric rectification can be seen.

Figure 21: Lines used for Metric Rectification

We normalized the lines as we only want two components (we divide the lines by the third one), we applied *Cholesky* to the matrix S, which follows the equation showed in figure 22, in order to find the matrix K that will allow us to metrically rectify the image.

$$H_a = \begin{pmatrix} A & \vec{0} \\ \vec{0}^T & 1 \end{pmatrix}$$

$$S = AA^T$$

Figure 22: Equivalence of Matrix S, Ha matrix from the Affinity Rectification

The results of the Metric Rectification can be seen in figure 23 and with the lines used in order to compute it in the figure 24

Figure 23: Result after Metric Rectification

Figure 24: Result after Metric Rectification with the computed lines

Finally, the angles we obtained between the lines are the ones showed in the table 2

Crossing Point Angles °	Yellow Lines	Red Lines
Previous	72.6	96.9
Transformed	87.2	85.6

Table 2: Angles between lines after Metric Transformation

5 OPTIONALS

5.1 Metric rectification from original image

As a first optional, we were asked to perform metric rectification from the original perspective image of the plane, not the affinely rectified image, based on [1]. We need to compute the H matrix. For doing so, we first must compute the conic C^*_∞, that it is determined on the perspectively image plane, in our case from the front wall of the building. Once we get the matrix C^*_∞ by *Single Value Decomposition* we are able to get the transformation matrix H, because $C^*_\infty = HC_\infty H^T$. By using the five orthogonal line pairs shown in figure 25 we are then able to obtain C^*_∞.

Figure 25: Orthogonal line pairs

The conic C^*_∞ determines the circular points, and equivalently the projective transformation necessary to metrically rectified the image.
As we can observe in figure 26, the result is not desirable.

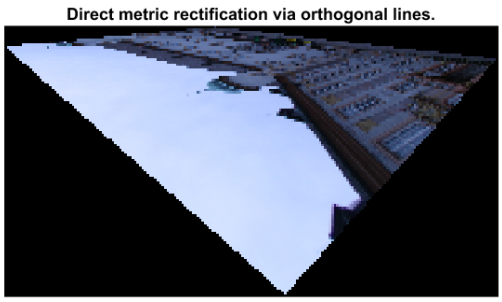

Figure 26: Orthogonal line pairs result

For computing the metric rectification we are selecting 5 pairs of lines. The problem comes when we select pairs of lines that are parallel between them.

5.2 Affine and metric rectification of additional facades

In the optional part, we were asked to apply affine and metric rectification of additional facades. We have implemented it in both images (0000 and 0001).

For the first image, we picked the lines corresponding to the left facade, see figure 27.

Figure 27: Lines used for Affine Transformation

Then we computed the Vanishing Points and the Vanishing Line. After this, we created our Affine Matrix. The result is shown on the figure 28.

Figure 28: Result after Affine Transformation

The angles we obtained are shown in table 3.

Angles º	Upper Left	Upper Right	Lower Left	Lower Rigth
Previous	57.73	116.13	63.84	110.03
Transformed	46.98	117.99	46.99	117.99

Table 3: Angles before and after the Affine transformation

Once the Affine Transformation is applied, we computed the Metric Rectification. Results can be seen in figure 29 and figure 30.

Figure 29: Result after Metric Rectification

Figure 30: Result after Metric Rectification with the computed lines

Finally, the angles we obtained between the lines are the ones showed in the table 4.

Crossing Point Angles º	Yellow Lines	Red Lines
Previous	117.99	54.72
Transformed	92.77	91.83

Table 4: Angles between lines after Metric Transformation

For the second image, we did exactly the same process. First we picked the lines corresponding to the left facade, see figure 31.

Figure 31: Lines used for Affine Transformation

Then we computed the Vanishing Points and the Vanishing Line. After this, we created our Affine Matrix. The result is shown on the figure 32.

Figure 32: Result after Affine Transformation

The angles we obtained are shown in table 5.

Angles º	Upper Left	Upper Right	Lower Left	Lower Rigth
Previous	66.85	112.14	71.23	107.72
Transformed	65.87	113.82	65.87	113.83

Table 5: Angles before and after the Affine Transformation

Once the Affine Transformation is applied, we computed the Metric Rectification. Results can be seen in figure 33 and figure 34.

Figure 33: Result after Metric Rectification

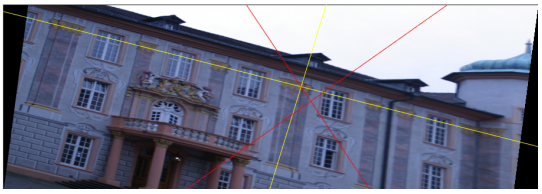

Figure 34: Result after Metric Rectification with the computed lines

Finally, the angles we obtained between the lines are the ones showed in the table 6.

Crossing Point Angles º	Yellow Lines	Red Lines
Previous	108.69	58.63
Transformed	87.13	88.58

Table 6: Angles between lines after Metric Transformation

6 CONCLUSIONS

After the first practicum, we have been able to see how the different images transformations allows us to reconvert our image in order to remove perspective, translate, rotate, etc.

As we can see from the Affine and Metric Transformations, we have been able to pass from an image with perspective distortion to a affine transformed image where the parallel lines are truly parallel from our view. The variation between the angles after the affine transformation allows us to see that the transformation is well done as there is almost no variation between them (72.7-72.6) compared with the variation that they had before the affine transformation (85.5-86.9).

After the affine transformation, applying a metric transformation allows us to recover the properties of the original image. As we can see from the results obtained, the angles between the different lines are closer to the 90^o objective than they were before (87.2 vs 72.6 and 85.6 vs 96.9). We can also conclude that it has not been possible to achieve a better result as we have supposed that the windows used to compute the lines follow a square form, which in reality may not be 100 per cent true.

REFERENCES

[1] R.I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.