M4 - 3D Vision

3D recovery of Urban Scenes

Ana Caballero - ana.caballeroc@e-campus.uab.cat
Arnau Vallvé - arnau.vallve@e-campus.uab.cat
Claudia Baca - claudiabaca.perez@e-campus.uab.cat
Joaquim Comas - joaquim.comas@e-campus.uab.cat

INTRODUCTION

The goal of this project is to learn the basic concepts and techniques to reconstruct a real world scene given several images (points of view) of it, not necessarily previously calibrated. In this project we focus on 3D recovery of Urban Scenes using images of different datasets, namely images of facades and aerial images of cities.

This project can be useful in different applications where some 3D information has to be inferred from images taken at different points of view. Examples of such kind of applications are: image mosaics or panoramas, augmented reality, depth computation, 3D reconstruction, 3D localization and navigation, new view synthesis.

WEEK 1

Goal

- How to transform an image by a homography. (applyH.m)
- The hierarchy of planar transformations. (Task 1)
- Affine rectification. (Task 2 + 4)
- Metric rectification. (Task 3 + 4)
- Metric rectification in a single step (Optional Task)

Task 0. Transform an image by a homography (apply_h)

The corners of the new image are precomputed so that we can automatically have the size that will be required beforehand. To do so, we transform the corners of the image in homogeneous coordinates using the homography and pick the bounding values so that we have the corners, which will be use to create a mesh of that size to fit the resulting image. Then we can safely compute the transformation of the image on the generated mesh by applying the inverse transformation in the projected space and interpolating the values of each coordinate in the image, so getting the new corresponding color for each point.

Task 1.1. Similarities

In this task we were asked to do a similarity transformation, to be able to do that we construct the next homography:

$$H_s = \begin{bmatrix} sR & \vec{t} \\ & \\ \vec{0} & 1 \end{bmatrix}$$

In where, as we have seen, *s* is a scaling factor, *R* is a rotation matrix and finally *t* is a translation vector. Below we can see the values that we use in our example.

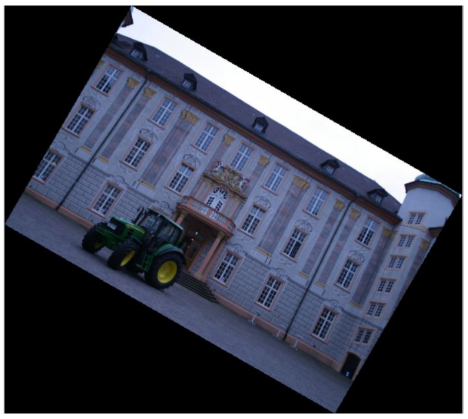
$$s = 0.5 \quad R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} = \begin{bmatrix} cos30 & -sin30 \\ sin30 & cos30 \end{bmatrix}$$

$$A = sR = \begin{bmatrix} s * cos\theta & -s * sin\theta \\ s * sin\theta & s * cos\theta \end{bmatrix} = \begin{bmatrix} 0.5 * cos30 & -0.5 * sin30 \\ 0.5 * sin30 & 0.5 * cos30 \end{bmatrix}$$

$$\vec{t} = \begin{bmatrix} 5 & 10 \end{bmatrix}$$

Then, applying this similarity transformation we get the next result:

Original image



Output image

As we can see in the output it is rotated 30º, scalated 0.5 and translated by t[5,10] and it is maintained the angles, the ratio of lengths and the ratio of two areas, so in other words the building has the same proportions.

Task 1.2. Affinities

In this next task, the aim was apply an affinity transformation, so apply to the image the next homography:

$$H_a = \begin{bmatrix} A & \vec{t} \\ \\ \vec{0} & 1 \end{bmatrix}$$

As before, for our example first we have to set up A, t and 0 in order to be able to calculate the homography matrix (3x3).In this case A is a non-singular 2x2 matrix and t it is still an a translation vector:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \qquad \vec{t} = \begin{bmatrix} 4 & 6 \end{bmatrix} \qquad H_a = \begin{bmatrix} A & \vec{t} \\ \\ \vec{0} & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 4 \\ 1 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

Then we called apply_H and plot the output.

Until here we have done the same as in Task 1.1, but it is in task we were also asked to decompose the affinity in four transformations: two rotations, a scale, and a translation. And then check if applying the decomposition we get the same transformation as before.

Here we have the three components of the decomposition:

- Rotations: We used SVD decomposition (*), to get the right singular vectors(Rsv) and left singular vectors (Lsv)

$$\text{rotation}\Phi = \begin{bmatrix} \text{Lsv} \cdot \text{Rsv}'(1,1) & \text{Lsv} \cdot \text{Rsv}'(1,2) & 0 \\ \text{Lsv} \cdot \text{Rsv}'(2,1) & \text{Lsv} \cdot \text{Rsv}'(2,2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{rotation}\Theta = \begin{bmatrix} \text{Rsv}'(1,1) & \text{Rsv}'(1,2) & 0 \\ \text{Rsv}'(2,1) & \text{Rsv}'(2,2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

-Scale: We used SVD decomposition (*), to get the singular values (sv) of A

$$\text{S} = \begin{bmatrix} \text{sv}\ (1,1) & \text{sv}\ (1,2) & 0 \\ \text{sv}\ (2,1) & \text{sv}\ (2,2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Translation:

$$T = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

(*) *https://es.mathworks.com/help/matlab/ref/svd.html#bu2_0hq-U*

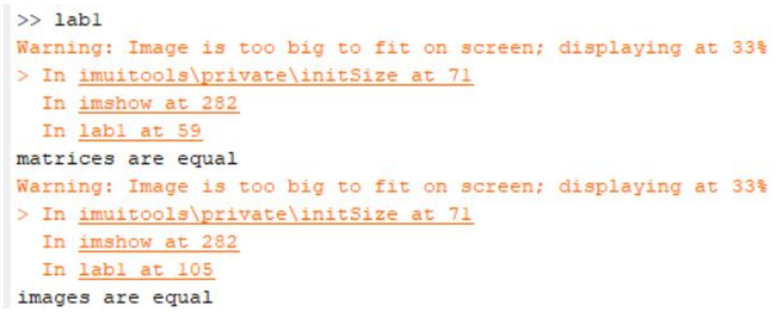
[U,D,V] = svd(A)

U = Left singular vectors D = Singular values V = Right singular vectors

Once we have all the components, we can compute the H_decomposition as a result of the next computation and apply it to the original image using apply_H:

$$H_d = T * (Rotation\theta * Rotation\phi * S * Rotation\phi)$$

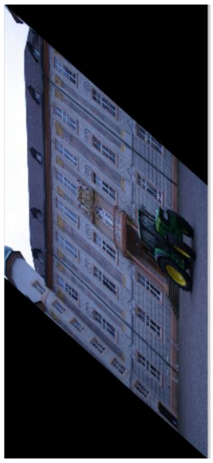
Then for the verification, we compute the difference between the H_d and H_a and then, the same with the two resulting images.



```
>> lab1
Warning: Image is too big to fit on screen; displaying at 33%
> In imuitools\private\initSize at 71
  In imshow at 282
  In lab1 at 59
matrices are equal
Warning: Image is too big to fit on screen; displaying at 33%
> In imuitools\private\initSize at 71
  In imshow at 282
  In lab1 at 105
images are equal
```

Results of the H_a:

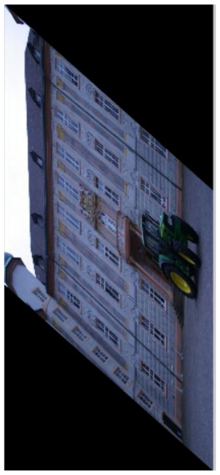
Original image



Output I2

Results of the H_d:

Original image



Output I2_decomposition

Also we can see visually that the two resulting images are equal, and that we get an affine transformation of the original image, where the parallelism is maintained, lines at infinity etc.

Task 1.3 Projective transformations

For this task we have to compute a projective transformation, so apply the next homography:

$$H_p = \begin{bmatrix} A & t \\ \vec{v} & vi \end{bmatrix}$$

As we know, a projective transformation can be decomposed in the next computation:

$$H_p = H_s * H_a * H_p = H_s * H_a * \begin{bmatrix} I & 0 \\ \vec{v'} & vi \end{bmatrix}$$

So we have reused the previous H_s and H_a to get a complete projective transformation only defining v and vi, and maintaining continuity in this tasks. Then we get the next Hp:

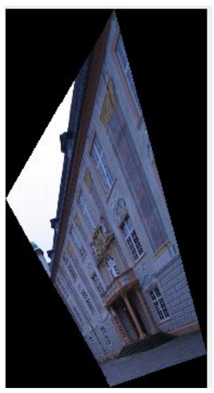
$$H_p = \begin{bmatrix} sR & t \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} A & \vec{t} \\ \vec{0} & 1 \end{bmatrix} * \begin{bmatrix} I & 0 \\ \vec{v'} & vi \end{bmatrix} =$$

$$= \begin{bmatrix} 0.5*cos30 & -0.5*sin30 & 5 \\ 0.5*sin30 & 0.5*cos30 & 10 \\ \vec{0} & & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 4 \\ 1 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.001 & 0.001 & 0.7 \end{bmatrix}$$

Where : $vi = 0.7$ $\vec{v} = \begin{bmatrix} 0.001 & 0.001 \end{bmatrix}$

And then we have applied the homography to the original image and we have get the next result:

Original image

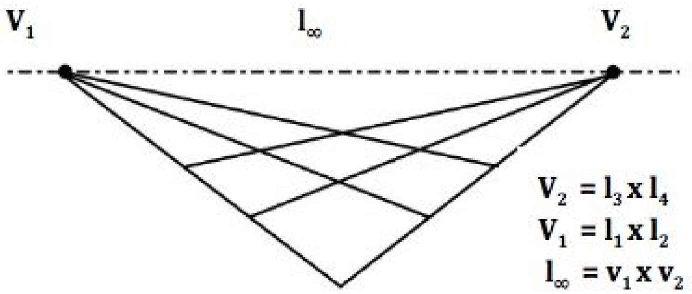

Output I2

As we can see in the resulting image there is not preservation of the angles, parallelism only it is maintained the coolinearty, concurrency, order of contact...

Task 2. Affine Rectification

The aim of the affine rectification is to remove the projective distortion of an image. Therefore, what we want is to recover some properties as the parallelism of the elements in the real world in our image 2D.

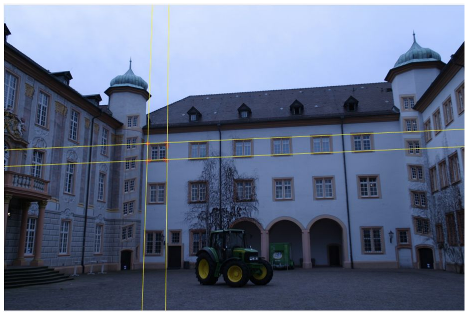
To restore our image we apply a homography to obtain a new image, where the vanishing line of the plane becomes the line in the infinity:



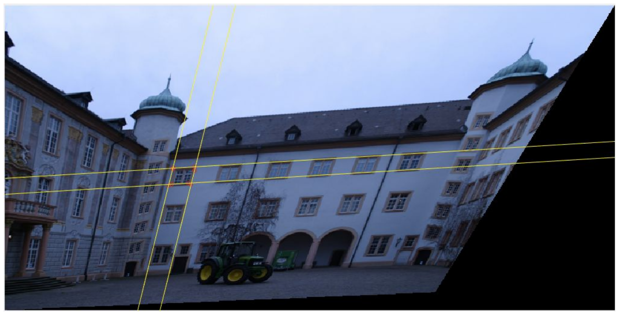
In more detail, in order to compute our homography, first of all we selected 8 points to compute 2 pair of parallel lines (as we can see in the original image). Then, we computed the 2 vanishing points (the point where two parallel lines intersect) and with these two resulting points we could obtain the vanishing line (line in the infinity). Finally, the resulting homography to rectify our image is:

$$H_{a<-p} = H_a \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{pmatrix}$$

Applying the resulting homography in the original image and original lines we obtained as following results:



Original image



Resulting image after affine rectification

Computed Angles obtained (before and after affine rectification):

```
Angle between l1 and l2 is , -0.099182 degrees
Angle between affine rectified lr1 and lr2 is , 0.000000 degrees
Angle between l3 and l4 is , 1.343537 degrees
Angle between affine rectified lr3 and lr4 is , 0.000000 degrees
```

From the computed angles between the parallel lines before and after rectification we can see how previously we had nearly parallel lines, with low angles near 1 deg. from each other, and how this angle has been reduced to 0 deg. for both line pairs l1||l2 and l3||l4 as they are now truly parallel to each other meaning the affine rectification procedure was a success.

Task 3. Metric Rectification after the affine rectification

The aim of the metric rectification is to recover metric properties, such as angles and length ratios from a previously affinely rectified image. What it is used is two pairs of affinely rectified lines, *l* and *m,* which are orthogonal on the world.

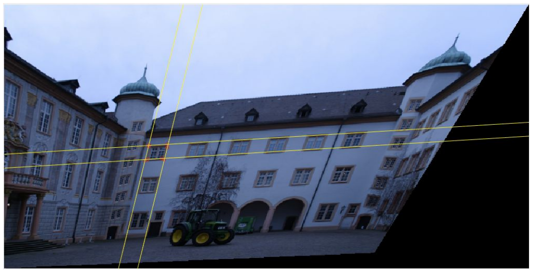
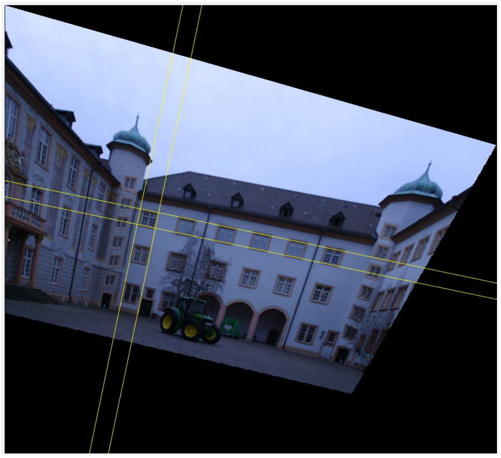
We want to impose the constraints that the lines *l* and *m* are orthogonal $(l_1m_1, l_1m_2 + l_2m_1 l_2m_2)\vec{s} = 0$. Using those two pairs of orthogonal lines the affine indeterminacy represented by:

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

can be restored by solving the system of equations from $S = AA^T$ and applying Cholesky decomposition to obtain the matrix K, which will give us our rectification matrix we wanted.

$$H_{a\leftarrow s}^{-1} = H_{s\leftarrow a} = \begin{pmatrix} K^{-1} & \vec{0} \\ \vec{0} & 1 \end{pmatrix}$$

Below we can see the images obtained when applying first the affine rectification and then the metric rectification. The metric properties have been restored. We will also see the numerical results of the angles between the orthogonal lines to ensure they are truly perpendicular.



Left. Image after affine rectification.

Right. Image after affine rectification + metric rectification.

Computed Angles obtained (before and after metric rectification):

```
Angle between l1 and m1 is , 72.640821 degrees
Angle between affine+metric rectified l1 transf. and m1 transf. is , -90.000000
degrees
Angle between l2 and m2 is , 72.640821 degrees
Angle between affine+metric rectified l2 transf. and m2 transf. is , 90.000000
degrees
```

This results show how before the procedure the lines were not perpendicular (they were angled approximately 73 degrees from each other) and how we were able to restore their angle to 90 deg. for both line pairs l1,m1 and l2,m2 meaning that the metric rectification procedure was successful.

Task 4. Affine and Metric Rectification of the left facade of image 0001

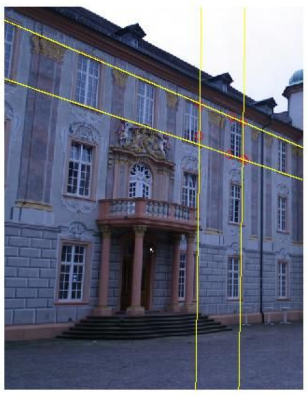
Similarly to the previous task, we will apply affine and metric rectification on a different image on which the left facade will be cropped to isolate it.

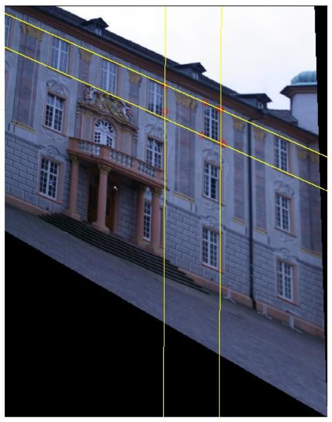
Original image with two facades

Cropped image showing only the left facade, which will be used on task 4

Without repeating too much, we will follow the two step method; we will first need to perform affine rectification to the image as seen in previous tasks given the adequate points and once affinely rectified the image we will perform the metric rectification on it.

Task 4.1. Affine rectification (left facade of image 0001)


Cropped image (left facade)

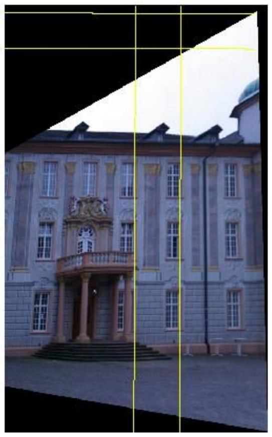

Affinely rectified image

Computed Angles obtained (before and after affine rectification):

```
Angle between l1 and l2 is , -4.420805 degrees
Angle between affine rectified lr1 and lr2 is , 0.000000 degrees
Angle between l3 and l4 is , 0.124941 degrees
Angle between affine rectified lr3 and lr4 is , 0.000000 degrees
```

After affine rectification the lines that were close to being parallel in the image have become truly parallel with angles of 0 deg. meaning the procedure was successful.

Task 4.2. Metric rectification after affine rectification (left facade of image 0001)



Metric + Affinely rectified left facade image

Computed Angles obtained (before and after metric rectification):

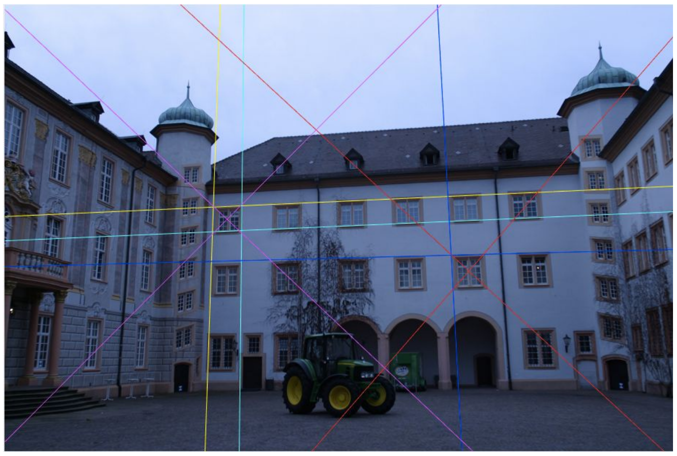
```
Angle between l1 and m1 is , -66.175047 degrees
Angle between affine+metric rectified l1 transf. and m1 transf. is , 90.000000
degrees
Angle between l2 and m2 is , -66.175047 degrees
Angle between affine+metric rectified l2 transf. and m2 transf. is , -90.000000
degrees
```

When applying metric rectification after affine rectification of the cropped left facade image we can see numerically and visually how the perpendicularity has been restored. A small issue however seems to appear when displaying the lines, but the results seem accurate.

Task 5 (Optional). Metric Rectification in a single step

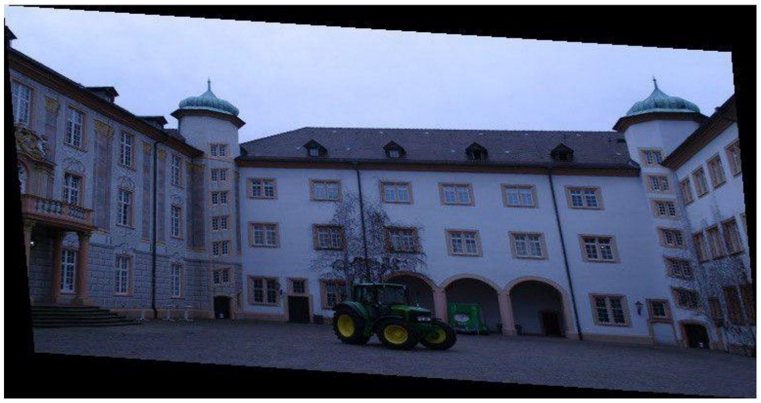
As in the previous section, the goal in this optional task is the metric rectification of an image, but in this case in a single step.

In this case, we used the Conic C_∞^*, which determines the circular points, and equivalently the projective transformation necessary to metrically rectified the image. In order to find the Conic C_∞^* we selected 5 pair of orthogonal lines as we can see in the next figure:



Selected pair of orthogonal lines

Then, using our 5 pairs of orthogonal lines we obtained a linear system of equations (5x6 matrix) $\mathbf{l}'^{\mathbf{T}} \mathbf{C}_\infty^* \mathbf{m}' = 0$, where $C_\infty^*$is obtained as the null vector. Finally, we applied the SVD to C_∞^* in order to obtain the transformation matrix H which rectify our image.



Result image after metric rectification in a single step

Conclusions:

- **Similarities:** maintains the proportions (angles and the ratio of lengths and areas)
- **Affinities:** the output images is equal to the original one as the parallelism is maintained, lines at infinity etc.
- **Projective transformation:** the resulting image does not preserve angles or parallelism but maintained the coolinearty, concurrency, order of contact, etc.
- **Affine Rectification:** angles between lines have been reduced for the pair of lines l1-l2 and l3-l4. Now they are parallel. This means affine rectification works as expected.
- **Affine + Metric Rectification:** we were able to restore the angle of the lines to 90º for the pair of lines l1-m1and l2-m2. This means metric rectification works as expected.
- **Affine + Metric Rectification (cropped facade):** Affine worked as well and the metric perpendicularity has been restored and the results look correct. But, we have and issue displaying the lines.
- **Metric Rectification (1 step):** In this optional part we had to change some parts of original algorithm, specially, the Cholesky decomposition by the SVD because in many cases we could not create a positive definite matrix using Cholesky decomposition. Respect to the result image, we can appreciate that it is different to the metric rectification into two steps, one possible reason to explain this difference could be the initial parallel lines used to perform the metric rectification.