Image Analysis and Computer Vision Homework

Federico Innocente

AY 2020/2021

Contents

1 Introduction

Figure 1: Castello di Miramare

The assignment is to analyze and image of the Castello di Miramare in Trieste (Figure 1). Some information about the building are provided: it is known that the line 1 and line 2 are orthogonal, as well as the lines 4 and 5 and lines 5 and 6 (hence), lines 4 and 6 are parallels. This information can be trivially extended to the whole facades that contains the lines. The tasks of the homework will be:

- Extract the needed features from the image
- Rectify the π plane, i.e. the plane containing the 6 lines
- Calibrate the camera
- Localize the camera with respect to π
- Rectify a vertical facade

2 Feature extraction

First of all, the image is preprocessed in order to increase the contrast of the image and better detect line. Than the canny algorithm is applied to get the image of the edges. The algorithm has been run two different times, one for the horizontal lines (Figure Figure 2) and one for the vertical once (Figure), and the three parameters of the algorithm have been empirically fine tuned to find the most useful lines.

The second step is to apply the hough space algorithm in order to find the edges of the image. The hough algorithm has been run on the respective edge detection image (Figure 4 for the horizontal and Figure 5 for the vertical),

and the parameters has been again manually tuned to exploit the best possible solution.

From the detected edges I extracted families of parallel lines (Figure 6 and Figure 7), by selecting as much as possible lines for each family, but avoiding lines too close to each other, that could have easily introduced high errors. Than all the lines has been homogenized and normalized, to have a better visual reference and to reduced the numerical errors due to the size of the image.

Figure 2: Edge detection of horizontal lines using canny algorithm

Figure 3: Edge detection of vertical lines using canny algorithm

Figure 4: Hough algorithm for horizontal lines

Figure 5: Hough algorithm for vertical lines

Figure 6: Horizontal lines selected

Figure 7: Vertical lines selected

3 Rectification of the plane π

This task has been done in two steps: first the image has been transformed with an affine rectification, and than an affine to metric transformation has been applied.

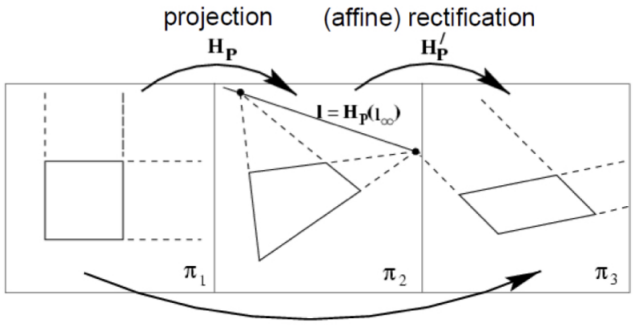

Figure 8: Two step rectification

3.1 Affine Rectification

The affine transformation is in the form

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ l1 & l2 & l3 \end{vmatrix}$$

where

$$l'_{\text{inf}} = [l1, l2, l3]$$

is the image of the line at infinity relative to the plane that we want to rectify. Since we want to rectify the plane π, all the lines parallel to it can be used to calculate the image of the line at infinity. The line at infinity l'_{inf} can be computed as the line that pass through the image of the vanishing points, that is the intersection of the parallel lines, but this method is would have been too noisy and too dependents by the selected lines. To avoid that, i selected as many families of parallel lines as possible, each containing several lines. Than i computed, for each family, all the possible intersections (Figure 11) and computed the vanishing point as the average intersection point (which projection on the computed image of the line at infinity is showed in figure 10).

Than, to compute the line at infinity, i calculated the line that better fitted the vanishing points (i.e. the line that better reduced the least square error), that is graphically showed in Figure 10.

Than i used this transformation to produce the affine rectified image and, as showed in Figure Figure 11, the parallels lines are correctly rectified (see lines

4 and 6).
The image of the line at infinity that i calculated is

$$l'_{\text{inf}} = [-0.012155952150397, -1.138901425521338, 1]$$

Figure 9: Lines intersection

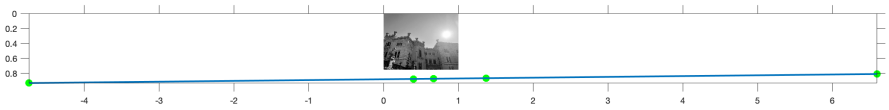

Figure 10: Image of the line at infinity w.r.t. plane π

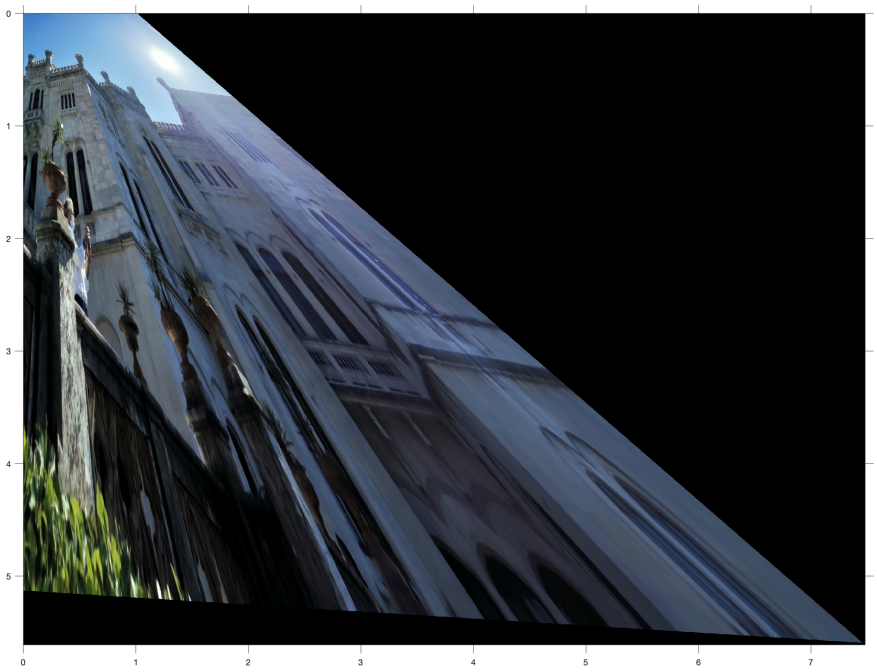

Figure 11: Affined image

3.2 Metric Reconstruction

The relationship between the original conic dual to the circular points $C^*_{\inf}$ and the affinely rectified image of the conic dual to the circular points $C'^*_{\inf}$ is:

$$C'^*_{\inf} = H_A C^*_{\inf} H_A^T = \begin{vmatrix} K & \mathbf{t} \\ \mathbf{0} & 1 \end{vmatrix} * \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{vmatrix} * \begin{vmatrix} K^T & \mathbf{0} \\ t^T & 1 \end{vmatrix} = \begin{vmatrix} KK^T & 0 \\ 0 & 1 \end{vmatrix}$$

From the angle constraints

$$\cos\theta = \frac{l'^T C'^*_{\inf} m'}{\sqrt{l'^T C'^*_{\inf} l' m'^T C'^*_{\inf} m'}}$$

we can retrieve $C'^*_{\inf}$. In particular, if l and m are perpendicoulars, we get that $l'^T C'^*_{\inf} m' = 0$. So I took the lines 1 and 2, and 5 and 6, and used them calculate $C'^*_{\inf}$ by applying singular value decomposition. Than, from

$$C'^*_{\inf} = H_A C^*_{\inf} H_A^T = \begin{vmatrix} K & \mathbf{t} \\ \mathbf{0} & 1 \end{vmatrix} * \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{vmatrix} * \begin{vmatrix} K^T & \mathbf{0} \\ t^T & 1 \end{vmatrix}$$

we get that

$$H_{rect} = H_A^{-1}$$

so

$$H_{rect} = \begin{vmatrix} K & \mathbf{t} \\ \mathbf{0} & 1 \end{vmatrix}^{-1}$$

The numerical result obtained is:

$$H_{rect} = \begin{vmatrix} 1.252341879710568 & -0.355501847405956 & 0 \\ -0.355501847405956 & 1.504317216297184 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

The graphical result (resized to solve a memory issue caused by Matlab) is showed in Figure 12. As can be seen, the orthogonal lines are turned so, like lines 1 and 2 or 4, 5 and 6

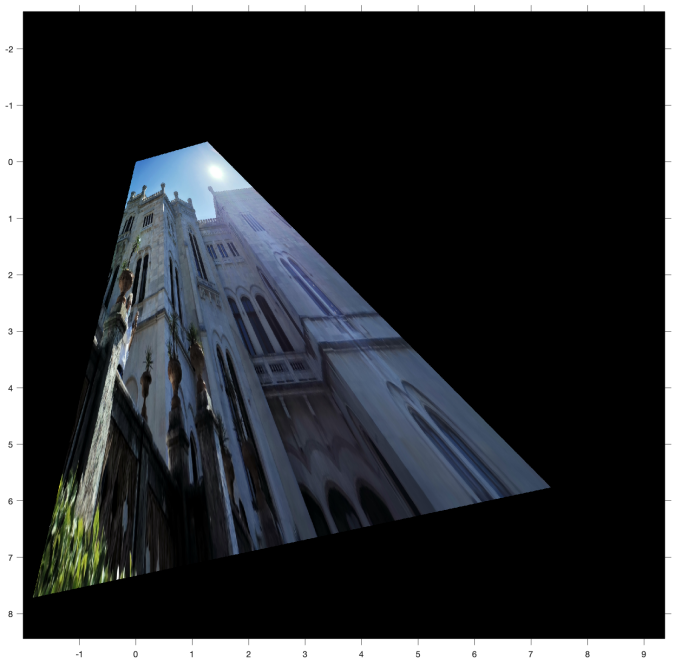

Figure 12: Metric rectified image

4 Camera Calibration

The camera calibration matrix is expressed as following:

$$\mathbf{K} = \begin{vmatrix} f_x & s & U_0 \\ 0 & f_y & V_0 \\ 0 & 0 & 1 \end{vmatrix}$$

The skew factor s is assumed to be 0, so the actual matrix that we want to compute is

$$\mathbf{K} = \begin{vmatrix} f_x & 0 & U_0 \\ 0 & f_y & V_0 \\ 0 & 0 & 1 \end{vmatrix}$$

The calibration give that

$$x = \mathbf{K} \begin{bmatrix} \mathbf{I}|0 \end{bmatrix} \begin{bmatrix} d \\ 0 \end{bmatrix}$$

so

$$\cos\theta = \frac{d_1^T d_2}{\sqrt{(d_1^T d1)(d_2^T d_2)}} = \frac{x_1^T (K^{-T} K^{-1}) x_2}{\sqrt{(x_1^T (K^{-T} K^{-1}) x_1)(x_2^T (K^{-T} K^{-1}) x_2)}}$$

We call

$$\omega = (KK^T)^{-1} = \begin{vmatrix} a^2 & 0 & -U_0 a^2 \\ 0 & 1 & -V_0 \\ -U_0 a^2 & -V_0 & f_y^2 + a^2 U_0^2 + V_0^2 \end{vmatrix}$$

The transformation of a direction is the image of the vanishing point, and if two of them are calculated over orthogonal lines

$$v_1^T \omega V_2 = 0$$

.

Since ω has 4 unknown, we need 4 independent equations to determine it, and since we want to exploit the easy calculus of the orthogonal vanishing points, I calculated the position of the vertical vanishing points (as in the previous steps, to reduce noise and dependencies i took the average intersection point among several vertical lines).

By solving the following equations

$$v_1 \omega v_2 = 0$$

$$v_5 \omega v_6 = 0$$

$$v_v \omega v_1 = 0$$

$$v_v \omega v_2 = 0$$

I obtained ω, and than by applying the Cholesky Decomposition on it I was able to find the camera calibration matrix K. The numerical results that I obtained are:

$$\omega = \begin{vmatrix} 1.101092044317959 & 0 & -2.265236774587324e+03 \\ 0 & 1 & -1.422382552031610e+03 \\ -2.265236774587324e+03 & -1.422382552031610e+03 & 1.555829319850057e+07 \end{vmatrix}$$

$$K = \begin{vmatrix} 2.839034649083077e+03 & 0 & 2.057263773974921e+03 \\ 0 & 2.979082328224971e+03 & 1.422382552031610e+03 \\ 0 & 0 & 1 \end{vmatrix}$$

Some considerations about the camera calibration:

- It can bee seen that $(KK^T)^{-1} = \omega$ except for a normalization factor. If we normalize for the element in position (2,2), that is supposed to be 1, the result is exactly ω.
- Differently form the rectification step, I worked with the original dimensions and not with normalized once, to obtain directly the calibration of the camera on the original image.
- It can bee seen that the parameters V_0 and U_0 are almost the half of the dimensions of the image, which can be considered a good result.
- Since $f_x \neq f_y$, the assumption of not having a natural camera was correct.

5 Camera Localization

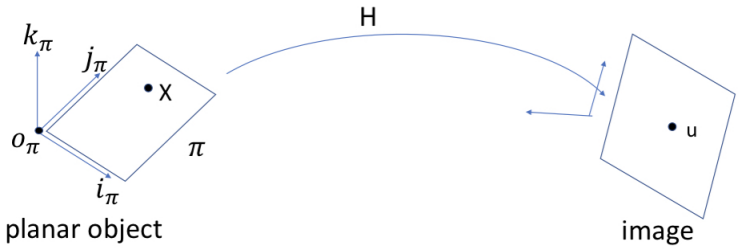

Figure 13: Localization model

A point $x_\pi = [xy0w]$ in the plane system reference is mapped to a point in the camera (or world) reference as

$$x_w = \begin{vmatrix} \mathbf{i}_\pi & \mathbf{j}_\pi & \mathbf{k}_\pi & \mathbf{o}_\pi \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 0 \\ w \end{vmatrix} = \begin{vmatrix} \mathbf{i}_\pi & \mathbf{j}_\pi & \mathbf{o}_\pi \\ 0 & 0 & 1 \end{vmatrix} x_\pi$$

Since from the camera calibration we know that, for a camera with reference on the world reference

$$P = \begin{vmatrix} KR & m \end{vmatrix} = \begin{vmatrix} K & 0 \end{vmatrix}$$

we can calculate that

$$u = Px_w = K \begin{vmatrix} \mathbf{i}_\pi & \mathbf{j}_\pi & \mathbf{o}_\pi \end{vmatrix} x_\pi$$

So the homography to take a point from the plane to the image is

$$H = K \begin{vmatrix} \mathbf{i}_\pi & \mathbf{j}_\pi & \mathbf{o}_\pi \end{vmatrix}$$

By inverting the transformation, we get that the pose of the planar object relative to the camera is

$$\begin{vmatrix} \mathbf{i}_\pi & \mathbf{j}_\pi & \mathbf{o}_\pi \end{vmatrix} = K^{-1}H$$

To calculate the transformation H, we need the positions o 4 points in the plane and camera references. Since we have no information about the dimensions of the castle, we need to work with relative positions. To get the camera points I plotted a rectangular on the roof of the castle's tower delimited by the lines 1 and 2. The first 3 points were the intersection of lines 1 and 2 and two random point on these two lines. To get the fourth point, I intersected line 1 and two with the image of the line at infinity to get the direction, and than used to draw the last two sides of the rectangular: the searched point was the intersection of

the two sides. To get the point in the plane reference, i just chose some standard coordinates: $[0, 0; -1 * s, 0; -1 * s, r * s; 0, r * s]$, where r is the ratio between the short side and the long side and s is a scale factor. The system reference of the plane has been set with i_π going from left to right and j_π going in the direction opposite to the camera. To know how to set the x, i transformed the points in the word reference to the metric rectified image by applying

$$x_{rect} = H_{rect} H_{aff} x_{image}$$

to calculate the ratio between the long and the short side of the rectangular, since here the ratio of the length are preserved (Figure 14).

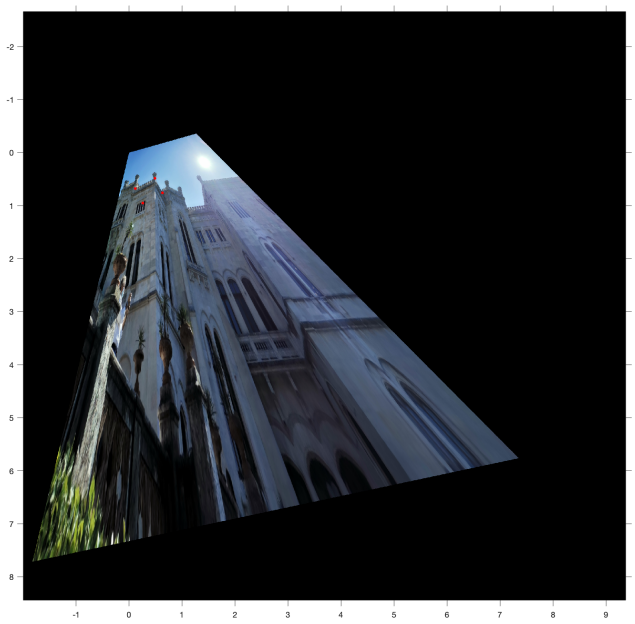

Figure 14: Localization points

Now I can compute H and than $\left|\mathbf{i}_\pi \quad \mathbf{j}_\pi \quad \mathbf{o}_\pi\right|$.

Once I obtain these results, I normalize the vector and calculate k_π by crossing i_π and j_π. The rotation matrix is than

$$R = \left|\mathbf{i}_\pi \quad \mathbf{j}_\pi \quad \mathbf{k}_\pi\right|$$

while the translation is

$$t = -Ro_\pi$$

The numerical results that I obtained are:

$$R = \begin{vmatrix} 0.981631753338433 & -0.114481299102825 & -0.152620879939286 \\ -0.152620879939286 & 0.553270329316560 & 0.810902754670881 \\ -0.008392596296190 & -0.825097554749543 & 0.564927950692594 \end{vmatrix}$$

$$t = \begin{vmatrix} 13.057162092701924 \\ -24.251617853873455 \\ -21.611929657683813 \end{vmatrix}$$

These results can be graphically visualized by plotting the relative positions of the points and the camera, as in the follow:

Figure 15: Reference points

As can be seen from Figure 16, assuming the facade 1 long 10 meters, the camera should be placed 20 meters under the roof of facade 1, at 20 meters of distance and 10 meters at the right, that, accordingly with the Figure 18 is a reasonable position.

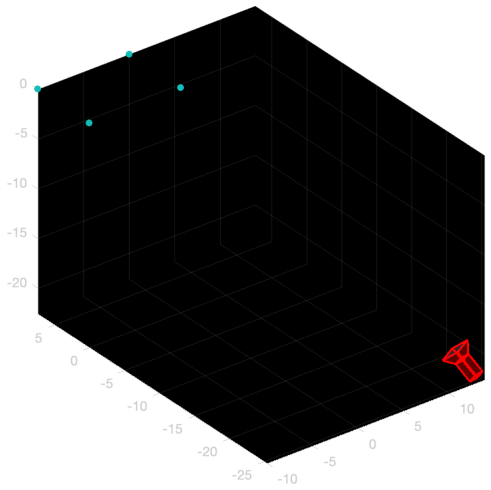

Figure 16: Camera localization, 1 unit is an estimate of 1 meter

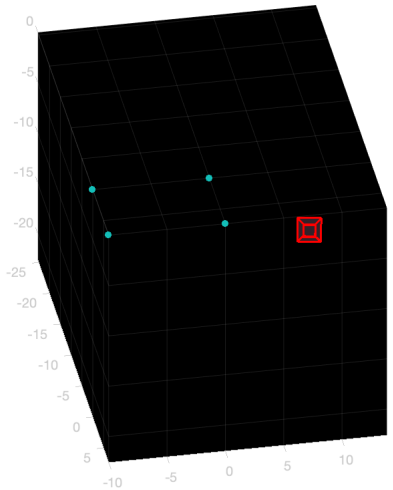

Figure 17: Camera localization, 1 unit is an estimate of 1 meter

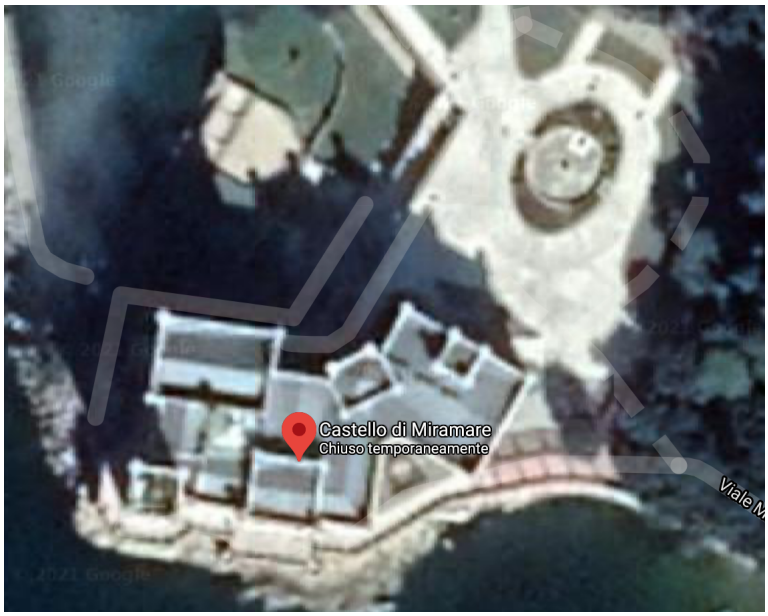

Figure 18: Visuale dall'alto del castello di Miramare con scala, foto presa da Google Maps

6 Frontal Rectification

Given the camera calibration matrix K, it is possible to calculate the image of the absolute conic ω as

$$\omega = (KK^T)^{-1}$$

Than, by performing the intersection between ω and the image of the line at infinity, we obtain the image of the circular points

$$l'_{\text{inf}} \cap \omega = \{I', J'\}$$

and, with the image of the circular points, it is possible to obtain the image of the conic dual to the circular points

$$C'^{*}_{\text{inf}} = I'J'^{T} + J'I'^{T}$$

By applying the singular value decomposition to C'^{*}_{inf} we get

$$svd(C'^{*}_{\text{inf}}) = U \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{vmatrix} U^T = H_{rect}^{-1} C^{*}_{\text{inf}} H_{rect}^{-T}$$

from which we can see that

$$H_{rect} = U^T$$

In my homework, I didn't computed ω by solving the product $(KK^T)^{-1}$, but I just reused the one computed for the camera calibration step. Than, to rectify the facade 1, I computed the image of the line at infinity as the line throw the vanishing points of lines parallels to 1 and the vertical once. The image of the circular points obtained by intersecting this two figures are:

$$I' = \begin{bmatrix} 6.988500924515598e+02 + 4.884278768077972e+03\text{i} \\ -2.853206870655873e+03 - 1.708677579691597e+03\text{i} \\ 1 \end{bmatrix}$$

$$J' = \begin{bmatrix} 6.988500924515598e+02 - 4.884278768077972e+03\text{i} \\ -2.853206870655873e+03 + 1.708677579691597e+03\text{i} \\ 1 \end{bmatrix}$$

that generate the image of the dual conic

$$C'^{*}_{\text{inf}} = \begin{vmatrix} 4.868914107203365e+07 & -2.067924301863962e+07 & 1.397700184903120e+03 \\ -2.067924301863962e+07 & 2.212073703619723e+07 & -5.706413741311746e+03 \\ 1.397700184903120e+03 & -5.706413741311746e+03 & 2 \end{vmatrix}$$

Than I applied the singular value decompositon

$$[U, D, V] = svd(C'^{*}_{\text{inf}})$$

which resulted in

$$U = V = \begin{vmatrix} -0.877633539394191 & -0.479332194357344 & 1.341007061691769e-04 \\ 0.479332208564949 & -0.877633458165379 & 3.833287448206541e-04 \\ -6.605054191745658e-05 & 4.007009507266739e-04 & 0.999999917538034 \end{vmatrix}$$

$$D = \begin{vmatrix} 5.998340784166608e+07 & 0 & 0 \\ 0 & 1.082647226656481e+07 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

from which I calculated the rectifying transformation

$$H = U * \sqrt{D} * V'$$

that, rescaled and inverted produced the rectification matrix

$$H_{rect} = \begin{vmatrix} 1.692794229734812 & 0.735348647608188 & 1.341007247650916e-04 \\ 0.735348647608188 & 2.637560072431442 & 3.833287979773366e-04 \\ 1.341007247650916e-04 & 3.833287979773367e-04 & 1.000000056209289 \end{vmatrix}$$

The transformation, applied to the image, give the result show in Figure 19, where can be seen that the edges are actually turned perpendicular.

By applying the same procedure to rectify the plane π (i.e. using the line at infinity calculated in Section 1) the graphical result is the one show in Figure 20, that, compared with Figure 12, show not only that the lines are actually transformed into orthogonal once, but also that this method have produced what it looks like the very same image.

Figure 19: Facade 1 Rectification

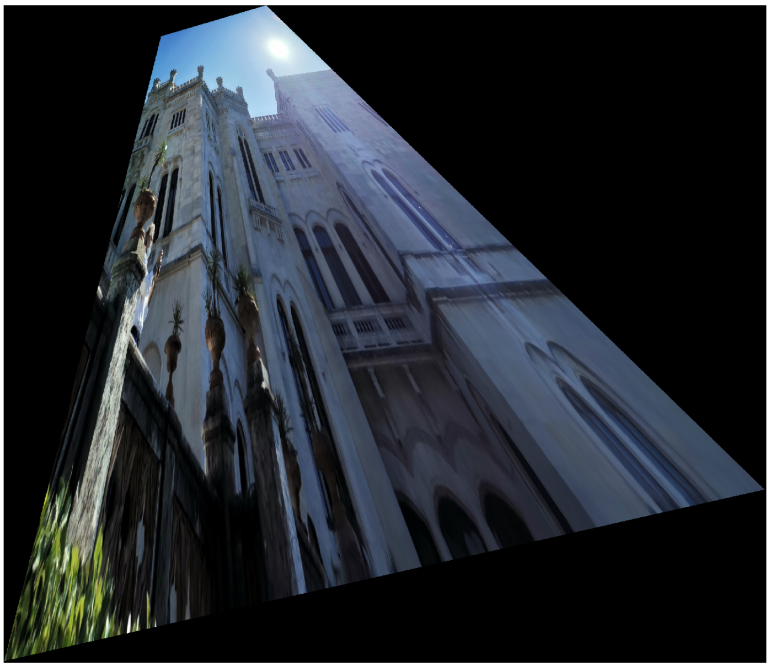

Figure 20: π Rectification using the calibration matrix