Image Analysis and Computer Vision Homework

Ghitti Marco

27/01/2020

Part I
Scene Description

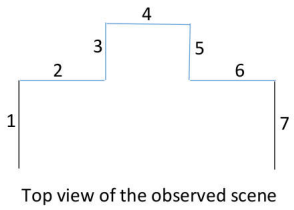

Figure 1: Top view

The observed scene contains a building showing a set of 7 vertical facades. Facades 1 and 7 are parallel and in front of eachother. Facedes 2 and 6 are coplanar, and perpendicular to facades 1 and 7. Facade 3 is parallel to facades 1 and 7, while the central facade 4 is parallel to facades 2 and 6. Facade 5 is occluded.

All the windows in facades 1, 2 and 6 are of equal width. The common width of the windows is 1 m. However, nothing can be assumed about the ornements regularly placed just under the roof cornice.

Figure 2: Image

Part II
Homework

1 Edges

The goal of this phase is to automatically find the edges of the image, this can be important to acquire important informations about the scene (Eg, object boundariers, textures, etc). In our case is also necessary for the line extraction section.

1.1 Solution

I decided to use the Canny edge detection algorithm, which consists of 4 steps:

1 - Gaussian smoothing: Since the edge detection algorithm is easily disturbed by image noise, is a good idea to smooth the starting image through convolution with a gaussian kernel; the sigma value of the gaussian distribution is chosen as sqrt(2) since is the default value in the matlab implementation of the canny edge detection algorithm.

$$S \coloneqq G_\sigma \otimes I \qquad\qquad G_\sigma = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$$

2 - Gradient magnitude and direction: The second step of the canny edge detection is to compute the gradient magnitude and direction.

To calculate magnitude and direction we need the partial derivatives with respect to the two main directions, x and y. This can be done by using any edge detection operator but the Sobel operator has been used since is the one that has been used on the slides.

$$\frac{d}{dx}S = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad\qquad \frac{d}{dy}S = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Once we have calculated the partial derivatives we can calculate gradient magnitude and direction as:

$$|\triangle S| = \sqrt{\left(\frac{d}{dx}S\right)^2 + \left(\frac{d}{dy}S\right)^2} \qquad\qquad \theta = \tan^{-1}\frac{\frac{d}{dy}S}{\left(\frac{d}{dx}S\right) + \epsilon}$$

where $\epsilon = 0.01$ is a small constant to avoid division by zero.

3 - Non maximum suppression: The non maximum suppression step check for each pixel if it is the maximum value along the direction of the steepest variation. If a pixel is not a local maxima along the gradient direction its value is suppressed.

$$M(x,y) = \begin{cases} |\triangle S|(x,y) & \text{if } |\triangle S| \text{ is a local maxima along } \theta \\ 0 & \text{otherwise} \end{cases}$$

4 - Hysteresis Thresholding: The goal of the Hysteresis Threshold step is to produce the final binary image by using two thresholds T_{HIGH} and T_{LOW} to eliminate false positives caused by noise and color variations.

The step declare a pixel as an edge pixel if it is above T_{HIGH} and a non edge pixel if it is below T_{LOW}. All other pixels are declared as edge pixels iff the pixel can be directly connected to an edge pixel or connected via pixels between the two thresholds.

1.2 Results

The results in figure show the comparison between the implemented edge detection algorithm and the Matlab edge method with canny as parameter.

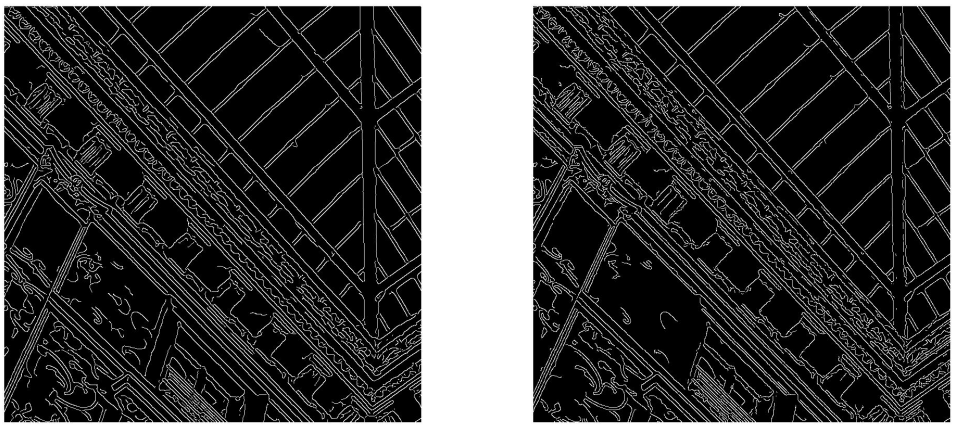

Figure 3: Edge detection results 1

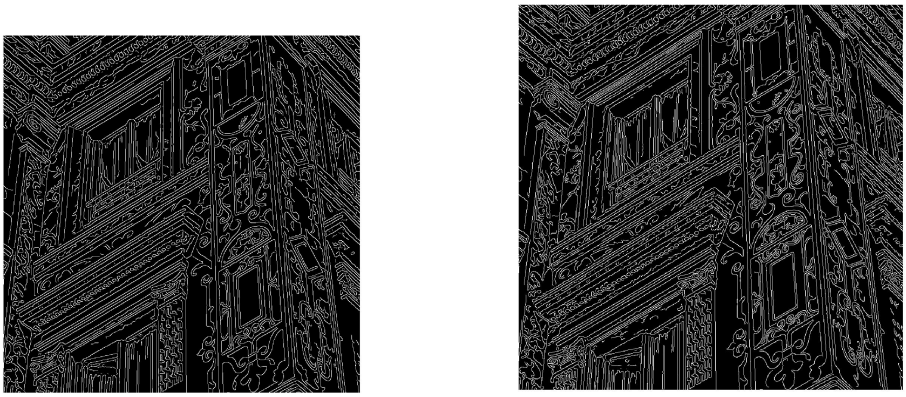

Figure 4: Edge detection results 2

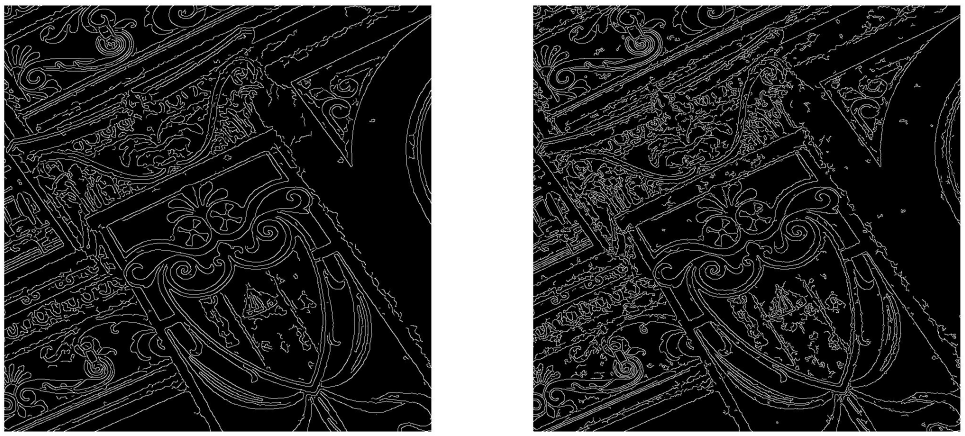

Figure 5: Edge detection results 3

2 Corner detection

The second step is to find the corners in the given image; this is done using the Harris corner detection algorithm.

2.1 Solution

The Harris corner detection algorithm consists of 4 steps:

1 - Spatial derivative calculation: The first step is to calculate, as in the Canny edge detection algorithm, the two partial derivatives with respect two the two main directions, x and y, with the Sobel operator.

2 - Calculation of the matrix $M_{r,c}$ Given the two derivatives we can calculate the matrix

$$M_{r,c} = \begin{bmatrix} I_x^2 \circledast w & I_x I_y \circledast w \\ I_x I_y \circledast w & I_y^2 \circledast w \end{bmatrix}$$

where w is a Gaussian filter.

The matrix $M_{r,c}$ have a well defined behaviour in corner points, in fact if the point (r, c) is a corner point it is also going to be a local minima of $M_{r,c}$ in all directions.

3 - Calculate CM To easily extract all points that are corners in the image we calculate a metric derived from the $M_{r,c}$ matrix.

$$CM = \frac{\det(M_{r,c})}{\mathrm{Tr}\,(M_{r,c}) + \epsilon}$$

4 - Local maxima of CM The last step is to compute the coordinate of all corner points by taking the coordinates that correspond to local maxima of CM.

2.2 Results

Comparison with the detectHarrisFeatures implementation of Matlab.

Figure 6: Corners detection results 1

Figure 7: Corners detection results 2

Figure 8: Corners detection results 3

3 Lines detection

The third feature that we need to compute are the lines inside the image; this can be done by using the results of the edge detection step and applying the Hough transform technique to find candidate lines.

3.1 Method

The Hough Transform Technique consist in chosing a parametrized family of models m, and implementing a voting mechanism to extract meaningful models.

The number of votes corresponding to a model is obtained as the number of data points that are compatible with m

$$V_X(m) = \#\{x_i \in X : f(m, x_i) = 0\}$$

and the candidate models are obtained as models that correspond to local maxima of V_X

$$\arg\max_m V_x(m)$$

It is worth considering that, due to noise in the image, continuous parameters must be discretized and that the function f, that measure the compatibility between model and data point, must instead consider points that are close to zero and not exactly zero.

$$f(m, x_i) \approx 0$$

Line Hough Transform: The Line Hough Transform is a special case of the Generalized Hough Transform; the considered model space is the model of a line parametrized as:

$$f(m, x_i) = X_i \cos(\theta) + Y_i \sin(\theta) - \rho$$

3.2 Algortithm

The Line Hough Transform algorithm can be dividedin 3 main steps:

1 - Parameter space definition and initialization: The parameters space corresponing to line models is a two-dimensional space with parameters θ and ρ; its definition must define range and granularity of both parameters.

$$\begin{aligned} \theta &= [-90°, +90°] & \text{Granularity} &= 1° \\ \rho &= [0, \text{maxImgSize}] & \text{Granularity} &= 1\text{px} \end{aligned}$$

where maxImgSize is the distance between the top left an the bottom right corners.

Before proceding to the next step the voting count of each cells is set to 0.

2 - Voting Procedure: Once the Hough Space has been defined we can start the voting procedure.

For each data point we consider its HT as the set of models m that are compatible with x_i

$$HT(x_i) = \{m \in \mathbb{R}^\kappa : f(m, x_i) \approx 0\}$$

Each data point add one vote to all models that are compatible with its value.

3 - Local Maxima extraction As in the General Hough Transform, the final step is to find the models that correspond to local maxima in the Hough Space; this is done by extensive search in the parameter space.

3.3 Results

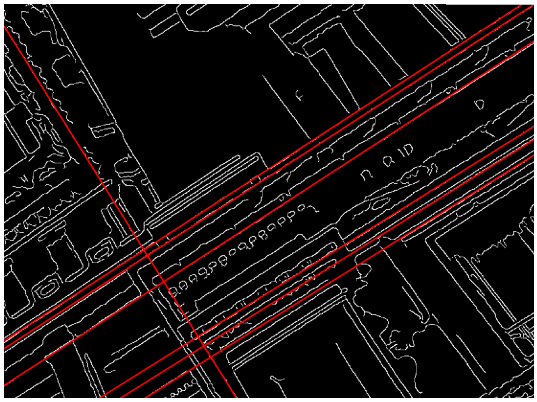

Figure 9: Lines detection 1

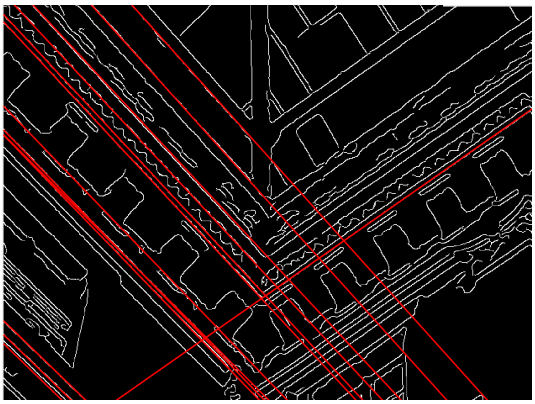

Figure 10: Lines detection 2

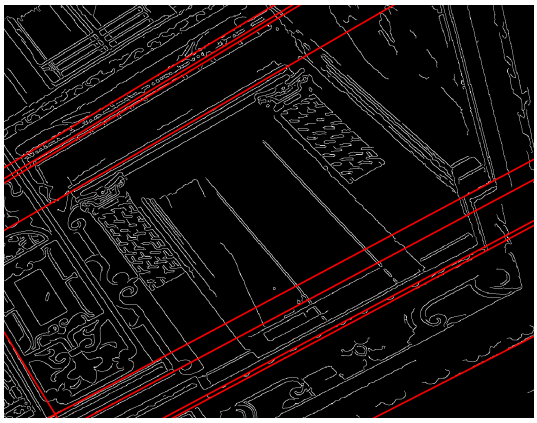

Figure 11: Lines detection 3

4 Geometry

4.1 Find two horizontal vanishing points and one vertical vanishing point.

Vanishing points are points on the image plane where the perspective projections of lines, which are parallel in the real scene, appear to converge in a single point.

Hence, to find two horizontal and one vertical vanishing points we need two set of horizontal parallel lines, corresponding to different horizontal directions, and one set of vertical parallel lines.

When more than two parallel lines are available, we can compute the final vanishing point as the centroid of the pairwise intersection of lines parallel in the real scene. This allow to better estimate the vanishing point corresponding to the direction in the real scene, and thus lead to better results for subsequent steps.

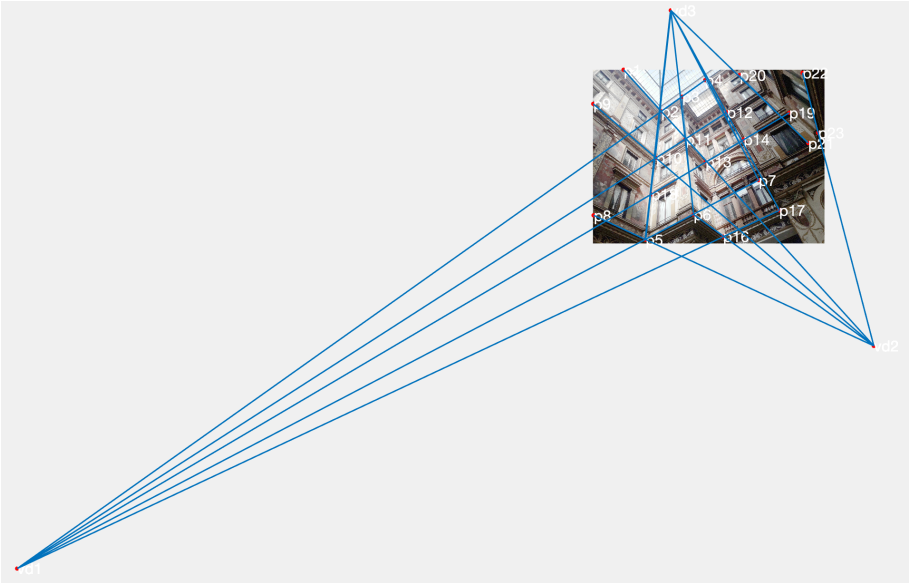

Figure 12: Vanishing Points

Given two points x_1 and x_2 in homogeneous coordinates, we calculate the line passing through the two points as

$$l = \text{cross}(x_1, x_2)$$

and the intersection point between two lines can be calculated as

$$v = \text{cross}(\text{l}_1, \text{l}_2)$$

4.2 Look at image below and consider the horizontal section of facades 1, 2 and 6 , depicted in yellow: metrically reconstruct this horizontal section*, so as to determine the relative coordinates of features points indicated in blue. These point features are placed (i) at the intersections between facades 1 and 2 and between facades 6 and 7, and (ii) in correspondence of borders of the windows.

The metric reconstruction can be performed in two steps to reduce numerical errors:

1 - From Projective to Affine The first step consist in finding a projective mapping H_P that maps the scene to an affine reconstruction with respect to the real scene. Since we already have two horizontal vanishing points we can calculate the line at the infinity l_∞ as the line that goes through the two horizontal vanishing points.

$$l_\infty = \text{cross}(v_{H_1}, v_{H_2})$$

Given l_∞ we can find H_P as

$$H_P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_{\infty_1} & l_{\infty_2} & l_{\infty_3} \end{bmatrix}$$

By applying H_P to the starting image we obtain the result that lines which are parallel in the real scene are also parallel in the affine reconstruction.

2 - From affine to Metric To metrically reconstruct the horizontal section we need to reconstruct its shape; to do so we can notice that, by knowing that each window have a width of exactly 1 meter, we can find two sets of 4 points that correspond to two coplanar squares in the metrically reconstructed scene.

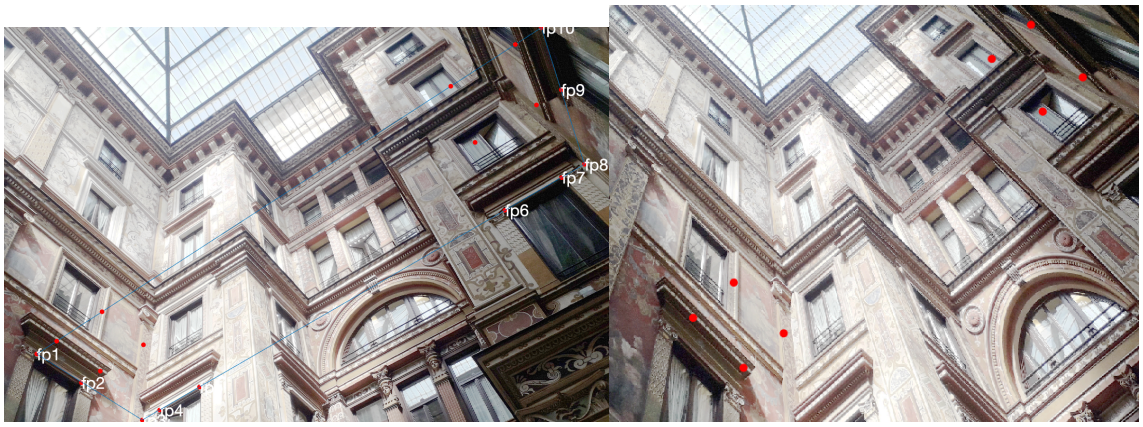

Figure 13: Windows shape

Hence, we can compute the homography H_R, that allow to upgrade the reconstruction from affine to metric, as the homography that map the 8 points in the affine reconstruction to two squares in the metric reconstruction.

The matrix H_R has been found with Least Squares Approximation to find the matrix that better reconstruct the shape of both squares.

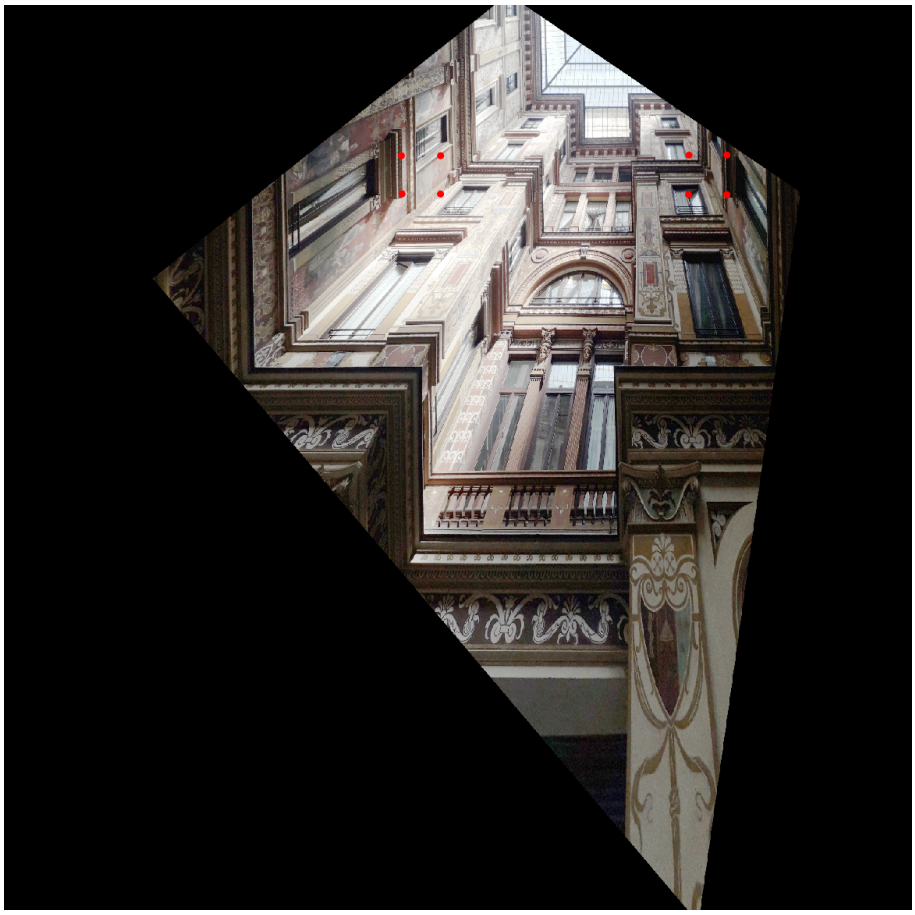

Figure 14: Metric Reconstruction

4.3 Estimate the calibration matrix K of the camera. Assume the camera is zero-skew, but not natural.

The calibration matrix K of the camera is the matrix that contain the intrinsic parameters of the camera; it's most general form is defined as

$$K = \begin{bmatrix} f_x & s & u_o \\ 0 & f_y & v_o \\ 0 & 0 & 1 \end{bmatrix}$$

The number of parameters can be reduced by making some assumptions on the camera parameters: A zero-skew camera is a camera with parameter $s = 0$, while a natural camera is a camera where $f_x = f_y = f$. Each restriction on the camera parameters reduce the number of degrees of freedom of the camera

matrix; hence, with the zero-skew assumption the calibration matrix have a total of 4 d.o.f.

The calibration matrix K can be computed by Cholesky factorization from the Image of the Absolute Conic that in our specific case can be parametrized as

$$\omega = (KK^T)^{-1} = \begin{bmatrix} \alpha^2 & 0 & -u_0\alpha^2 \\ * & 1 & -v_0 \\ * & * & f_y^2 + \alpha^2 u_0^2 + v_0^2 \end{bmatrix}$$

Hence, if we are able to find the IAC we can directly compute the calibration matrix K; to do so we need to apply 4 independent contraints to find its parameters.

Since we have already computed 3 vanishing points we can apply 3 constraints, one for each of pair of vanishing points corresponding to orthogonal directions

$$v_1^T \omega v_2 = 0$$

The last constraint can be obtained from the complete metric recostruction homography that can add 2 additional constraints

$$H = (H_R * H_P)^{-1} = [h_1, h_2, h_3]$$
$$h_1 \omega h_2 = 0 \qquad h_1^T \omega h_1 = h_2^T \omega h_2$$

The IAC is then fitted through Least Square Approximation and K can be found with Cholesky factorization.

```
1.0e+03 *

  3.2320         0    2.0172
       0    3.2474    1.3854
       0         0    0.0010
```

Figure 15: Matrix K

4.4 Use the knowledge of K to rectify also a vertical facade, as, e.g., facade 1 or 4 or 2+6.

To rectify a vertical facade we need to know the full camera matrix P; by using the extrinsic camera parameters from the next step, we can compute the camera matrix P as from definition

$$P = K[R|T] = [p_1|p_2|p_3|p_4]$$

We can then extract the rectifying homography H_{vert} as

$$H_{\text{vert}} = [p_1|p_3|p_4]^{-1}$$

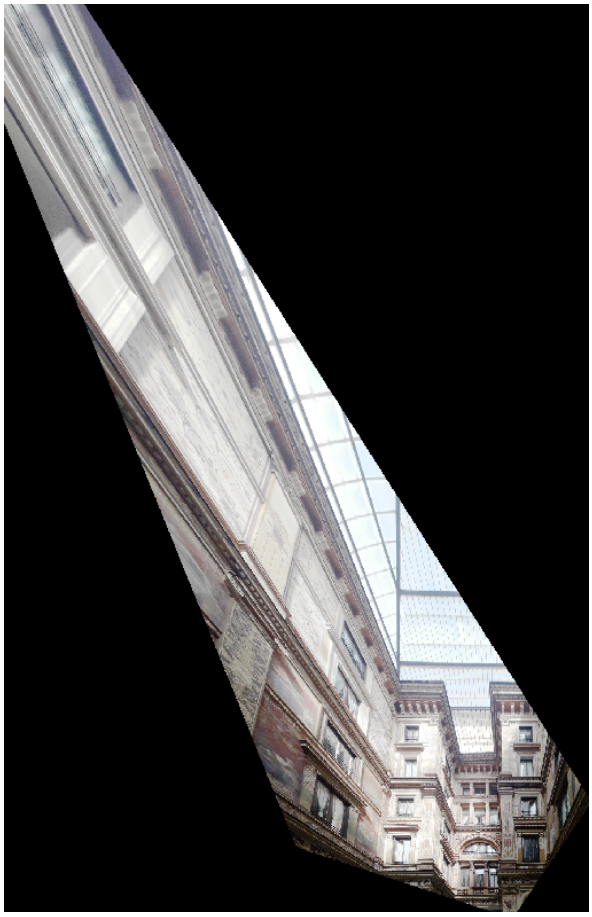

Figure 16: Vertical Rectification

4.5 Fix a suitable reference frame attached to the building, and localize the camera relative to the fixed reference

We can localize the camera by finding its relative position with respect to the left square on the metrically reconstructed plane. We know that

$$[i_\pi|j_\pi|o_\pi] = K^{-1}H_{\text{omo}}$$

where H_{omo} is the homography that map the shape of the square to the image.

The rotation matrix R can then be computed as

$$R = [i_\pi | j_\pi | k_\pi] = [i_\pi | j_\pi | i_\pi \times j_\pi]$$

and the translation vector as

$$T = o_\pi$$

```
cameraRotation =

    0.8281   -0.4983   -0.2567
    0.4390    0.6302    0.7116
   -0.1928   -0.7020    0.7407

>> cameraPosition

cameraPosition =

   1.0e+03 *

    1.2773
   -1.1016
   -1.2999
```

Figure 17: Rotation and Position