Image Analysis and Computer Vision

Homework 2021-2022

Mirko Usuelli (10570238) – mirko.usuelli@mail.polimi.it

January 25, 2022

Prof. Vincenzo Caglioti, Politecnico di Milano.

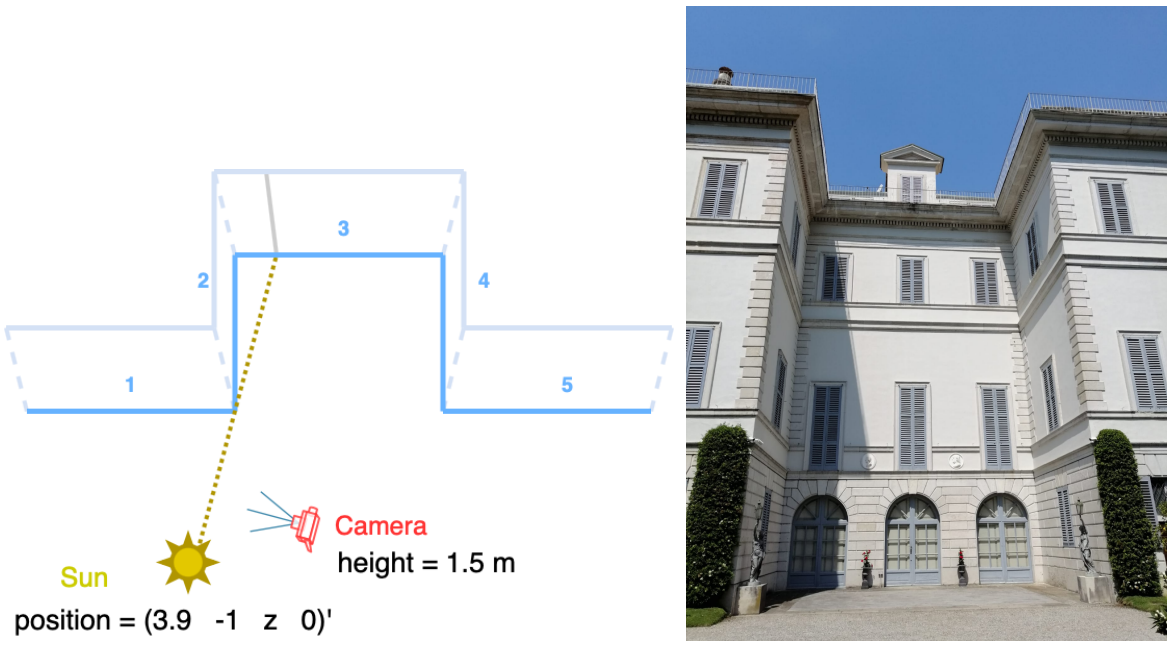


Figure 1: Homework Scene - Villa Melzi

Image Processing

F1 - Feature extraction

1. **Edges Detection**: I applied the Canny Edge Detector algorithm which allowed me to extract useful image edges by applying the following steps:

(a) Apply Gaussian filter to smooth the image in order to remove the noise

(b) Find the intensity gradients of the image

(c) Apply gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection

(d) Apply double threshold to determine potential edges

(e) Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Figure 2: Edges Detection

2. **Corners Detection**: In order to select coners I considered to select the Harris Corner Detector algorithm which allowed me to extract image corners. The algorithm consists of applying the following procedure:

   (a) Color to grayscale in order to work on a single channel and enhancing speed processing.

   (b) Spatial derivative calculation to highlight pattern changes around each pixel region analyzed and then setting up the tensor structure.

(c) Harris response calculation by computing the smallest approximated eigenvalue of the structure tensor.

(d) Non-maximum suppression in order to pick up the optimal values to indicate corners, I find the local maxima as corners within the window which is a 3 by 3 filter.

Figure 3: Corners Detection

3. **Lines Detection**: I decided to implement Hough Transform algorithm in purpose of extracting image lines by applying the following operations:

(a) Edge detection: using the Canny edge detector.

(b) Mapping of edge points to the Hough space and storage in an accumulator.

(c) Interpretation of the accumulator to yield lines of infinite length. The interpretation is done by thresholding and possibly other constraints.

(d) Conversion of infinite lines to finite lines based on the hyperparameters setting.

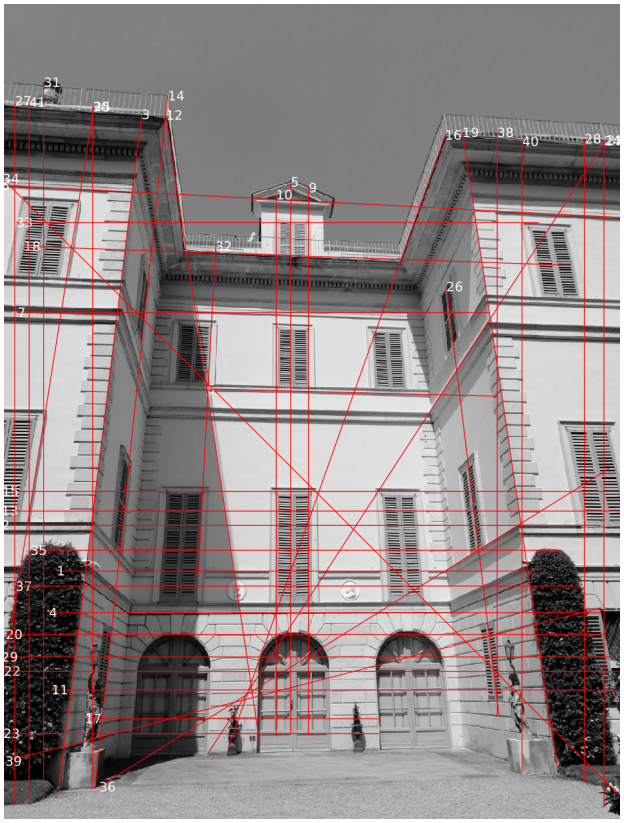


Figure 4: Lines Detection

Geometry

G1 - 2D reconstruction of a horizontal section

In order to reconstruct the horizontal section I focused on a stratified rectification approach due to the lack of constraints for a full metric rectification from the original image.

The stratified approach consists in two steps:

(a) **Affine Rectification** *(2 constraints)*: by selecting two independent pairs of parallel lines in the image scene, I am able to estimate their intersection by cross product and find the image of the line at the infinity linking those vanishing points. The purpose of the affine rectification is to bring back the image of the line at the infinity at its own canonical position while preserving and restoring lines parallelism as direct consequence of the original position of the line at the infinity.

$$H_{aff} = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \\ -0.0001 & -0.0008 & 1.0000 \end{bmatrix}$$

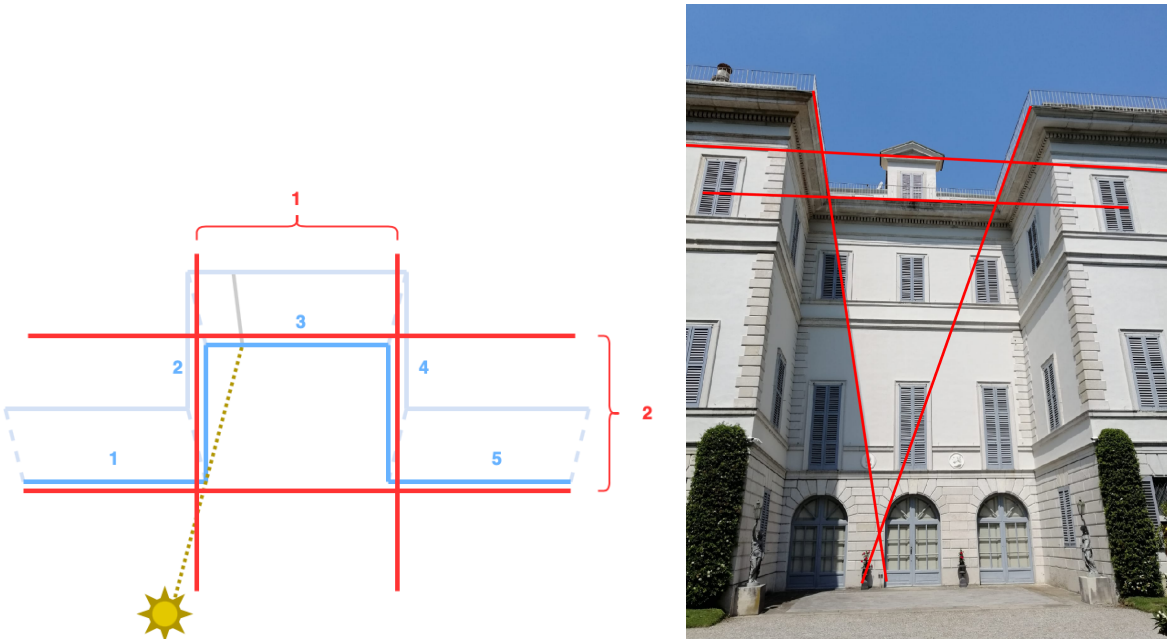


Figure 5: Affine constraints in the (1) scene and (2) image.

More in details, an affine transformation is a non-singular linear transformation followed by a translation. Its matrix representation is the following:

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

Due the fact that an affine transformation includes non-isotropic scaling, the similarity invariants of length ratios and angles between lines are not preserved under an affinity. Indeed, Affine Transformation invariants are: Parallel lines, Ratio of Areas, Ratio of Lengths of parallel segments.

An affine transformation maps points at infinity to points at infinity. Meaning that the parallel lines are mapped to lines which still intersect at infinity, thus parallel after the transformation.

In order to perform affine rectification I require that the line at infinite in the image is mapped back to itself ($l_\infty = (0, 0, 1)^T$).

(b) **Euclidean Rectification** *(2 constraints)*: due to the previous transformation I was able to reduce necessary constraints to find and bring back the Circular Points – with respect to the horizontal plane – to their canonical position. This is ensured by taking two independent constraints.

As first constraint I chose a pair of orthogonal lines along a building angle. Whereas, for the second constraint there was not a clear second pair of orthogonal lines independent with the first one. I decided to exploit the information about the sun position

(considered at the infinity) combined with a vertical shadow of the facade 3 which allowed me to discover a proportion similarities ensured by the Theorem of Similar Triangles as shown in the image below:

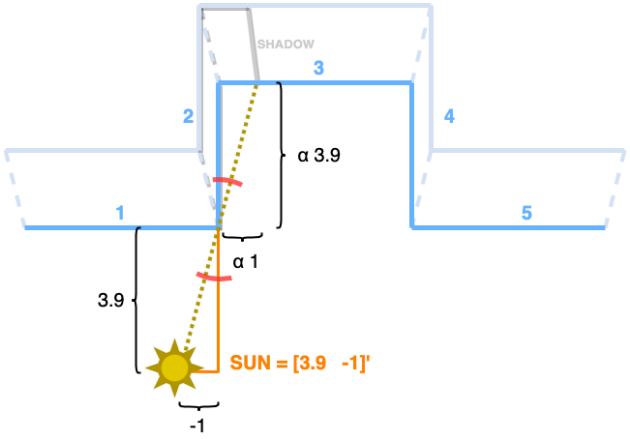


Figure 6: Similar Triangles

Therefore, I could use this important information to set a constraint on the Ratio of Lenghts ($k = 1/3.9$) of two lines – l and m – since the Euclidean Transformation preserves them; that, paired with the orthogonal lines constraint, is a linear constraint on $C^*_\infty{}'$.

$$\sqrt{(l'^T C'^*_\infty l')} = k \cdot \sqrt{(m'^T C'^*_\infty m')}$$

$$(l'^T C'^*_\infty l') = k^2 \cdot (m'^T C'^*_\infty m') \longrightarrow l'^T C'^*_\infty l' - k^2 \cdot m'^T C'^*_\infty m' = 0$$

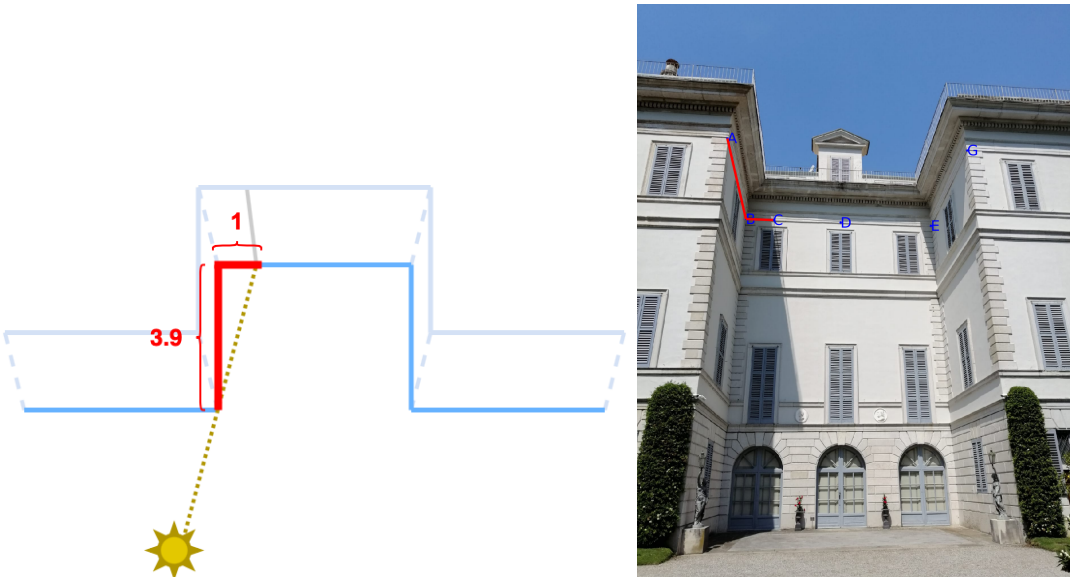


Figure 7: Ratio of Lenghts constraint in the (1) scene and (2) image.

More in details, a similarity transformation is an isometry composed with an isotropic scaling. In the case of a Euclidean transformation composed with a scaling – i.e. no reflection – the similarity has matrix representation:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

where the scalar s represents the isotropic scaling. A similarity transformation preserves the shape of objects, furthermore, it is characterized by the following invariants: Ratio of lengths, Ratio of areas, Angles between lines (and parallel lines).

Once the image has been affinely rectified, I have obtained an image such that the transformation from the original scene is an Affine Transformation:

Figure 8: Affine Transformation with the Euclidean constraints highlighted.

Given these two constraints, I am now able to perform the Euclidean Rectification. I exploited the SVD decomposition in order to solve the constraints and compute the image of the conic dual to circular points, then I inserted the solutions in a 2×2 matrix:

$$S = \begin{bmatrix} sol_1 & sol_2 \\ sol_2 & sol_3 \end{bmatrix} \tag{3}$$

Next, I applied again SVD decomposition of S obtaining the Metric Transformation coefficients:

$$U, D, V = SVD(S)$$

$$a = U \cdot \sqrt{D}$$

$$H_{met} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-T} = \begin{bmatrix} 1.2398 & -0.3136 & 0 \\ -0.3136 & 1.4399 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

Obtaining the final Horizontal Shape Transformation as: $H_{rec} = H_{met} \cdot H_{aff}$

Figure 9: Horizontal Rectification.

The requested ratio of facade 2 (or facade 4) over facade 3 is equal to: `0.6034`

G2 - Calibration

In order to calibrate the Camera it was necessary to select at least four constraints – since the zero skew camera was already a constraint itself – to get the Image of the Absolute Conic (IAC). The IAC is strongly related to the calibration camera matrix K through the relation $w = (KK^T)^{-1}$:

$$w = \begin{bmatrix} \alpha^2 & 0 & -u_0\alpha^2 \\ 0 & 1 & -v_0 \\ -u_0\alpha^2 & -v_0 & f_y^2 + \alpha^2 u_0^2 + v_0^2 \end{bmatrix}$$

I first computed a vertical vanishing point by intersecting two vertical lines of the building through the cross product. Then I selected two pairs of parallel lines relying on a horizontal plane as I did for point G1.

In this way I could find two vanishing points and then compute the image of the line at the infinity orthogonal to the previous vertical vanishing point. The fact of using the line at the infinity ensured us to have two constraints with the same equation, because it would be equivalent to have two separate constraints using orthogonal vanishing points alone.

$$[l_{inf}]_\times = v_{p1} \times v_{p2}$$

$$[l_{inf}]_\times w v_p = 0$$

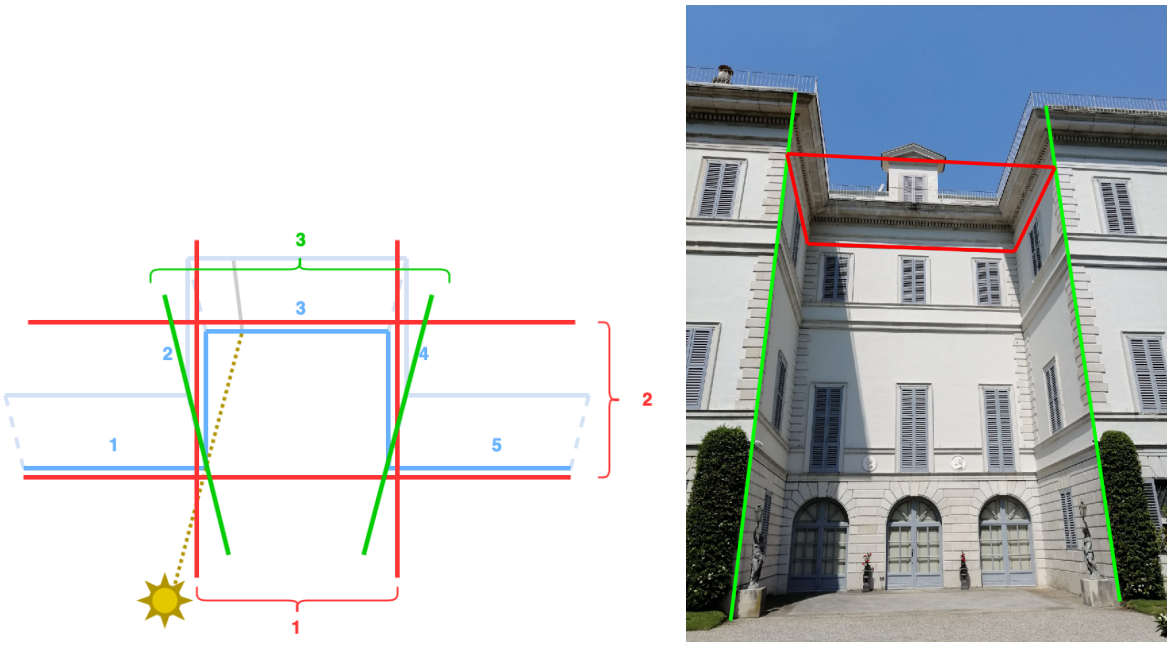


Figure 10: Calibration constraints in the (1) scene and (2) image.

Then the other two constraints could be easily recovered by exploiting the Rectify Transformation I have found at point G1 by using the homography $H = \left[h_1, h_2, h_3\right]$ combined with the image of Circular Points $h_1 \pm i \cdot h_2$:

$$h_1^T w h_2 = 0$$

$$h_1^T w h_1 - h_2^T w h_2 = 0$$

Obtaining the calibration matrix K through Cholesky Decomposition of $w = (KK^T)^{-1}$ as:

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1226.9 & 0.0 & 533.5 \\ 0.0 & 883.3 & 904.9 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Where $f_x, f_y, u0, v0$ are the intrinsic parameters of the camera.

G3 - Reconstruction of a vertical facade

Through the knowledge of the calibration matrix K found at point G2 I was able to compute the image of the absolute conic:

$$w = (KK^T)^{-1}$$

Then it was enough to compute again the image of the Line at the Infinity l'_{inf}, with respect to the vertical plane π I was going to rectify, and intersect it with the Image of the Absolute Conic.

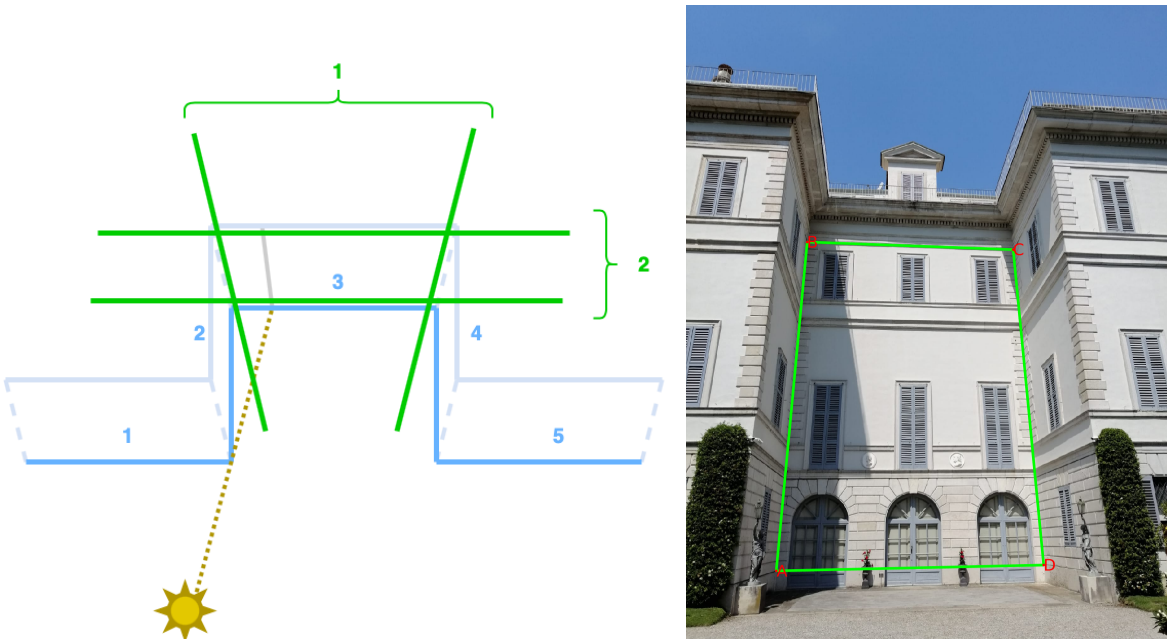


Figure 11: Vertical constraints in the (1) scene and (2) image.

The result led us to the Image of the two Circular Points as direct result of the equations.

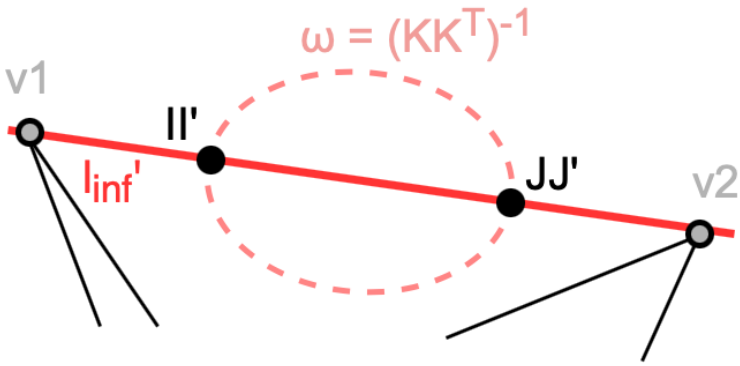


Figure 12: Image of Circular Points

Therefore I just needed to bring the Circular Points back to their canonical position in purpose of obtaining the desired vertical plane rectified as I did for the point G1 through SVD decomposition.

The ratio of the short side over the long one of facade 3 is equal to: `0.4956`

Figure 13: Before and After the vertical rectification.

G4 - Localization

In this last point I had to find the relative position of the camera with respect to the reference frame placed on the bottom left corner of the vertical face (facade 3). This was possible knowing the shape of the vertical face, knowing the height of the camera in order to scale proportions (1.5 meters expressed as 1.5 units for the sake of convenience), knowing the image and knowing the calibration matrix K.

I first identified the plane of the vertical facade 3 with π, then I could write the position of a point in the world reference frame as $X = [R_\pi|o_\pi]X_\pi$ where X_π was the position of the point in the plane reference frame. In this way a point on the plane could be written (in homogeneous coordinates) as:

$$X_\pi = \begin{bmatrix} 0 \\ y \\ z \\ w \end{bmatrix}$$

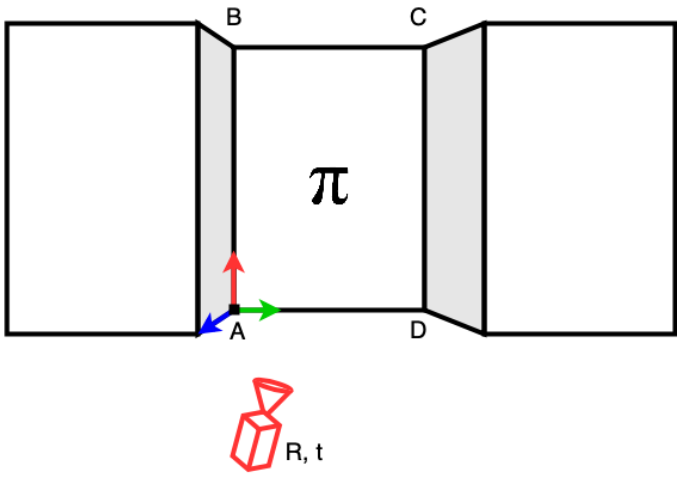


Figure 14: World reference frame

Knowing equations of the projection matrix P which maps 3D points (X) in the image (x):

$$x = PX$$

and

$$P = [KR| - KR_o]$$

Where R was the rotation between the camera and the world and o was the location of the camera in the world reference frame in cartesian coordinates; I could derive:

$$u = [K|0] \begin{bmatrix} j_\pi & k_\pi & o_\pi \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y & z & w \end{bmatrix}$$

$$u = K \left[j_\pi | k_\pi | o_\pi \right] x$$

Where x were the coordinates of the point on the plane. By inspection I could identify $K \left[j_\pi | k_\pi | o_\pi \right] = H$ as the matrix that maps the real points to the image. H was known since the shape of horizontal face was known and also its size; hence I obtained:

$$\left[j_\pi | k_\pi | o_\pi \right] = K^{-1} H$$

Where $H = \left[h_1 | h_2 | h_3 \right]$ was the transformation mapping world points to image points.

The rotation matrix $R = \left[i_\pi \mid j_\pi \mid k_\pi \right]$ of the plane π with respect to the camera could be computed through these equations:

$$\lambda = \frac{1}{|K^{-1} h_1|}$$

$$j_\pi = K^{-1} h_1 \lambda$$

$$k_\pi = K^{-1} h_2 \lambda$$

$$i_\pi = j_\pi \times k_\pi$$
$$o_\pi = K^{-1} h_3 \lambda$$

Due to the noise in the data, R might be not a true rotation matrix, thus I approximated it through SVD, obtaining an orthogonal matrix:

$$U, \sim, V = SVD(R)$$
$$\hat{R} = U \cdot V$$

The following images clearly show – from different perspective – that camera localization scaled with the known camera height in the 3D space had success; whereas the camera position – expressed in meters – is:

$$\texttt{cameraPosition} = \begin{bmatrix} 19.5023 \\ 2.8950 \\ 1.5000 \end{bmatrix}$$

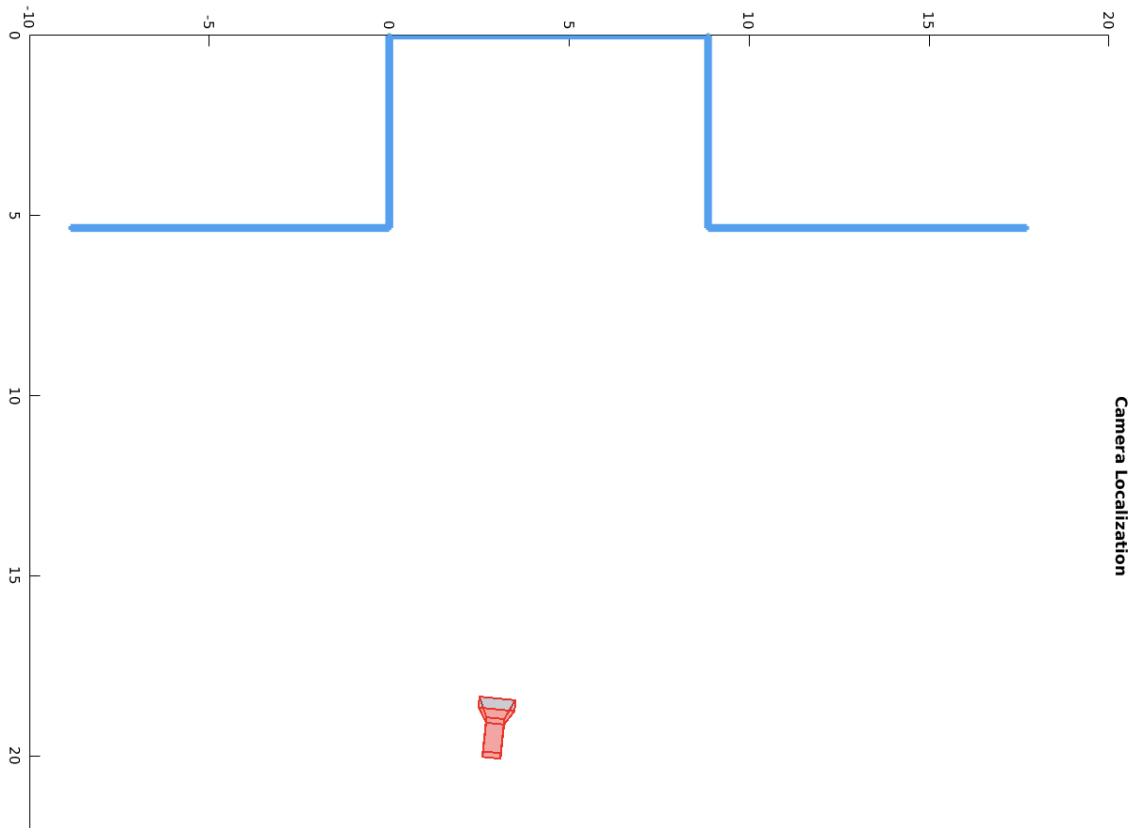


Figure 15: Top Scene in the 3D space.

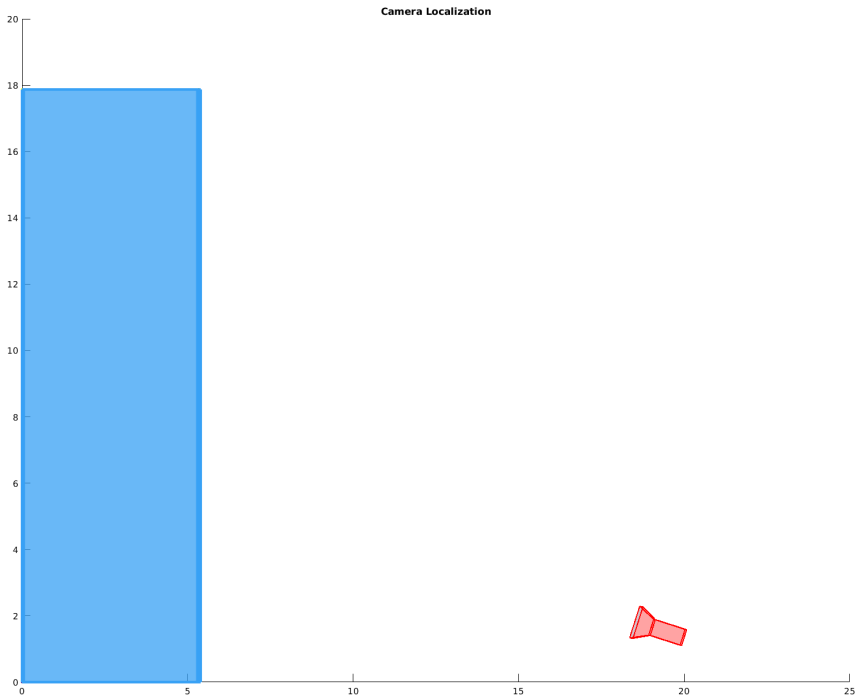


Figure 16: Lateral Scene in the 3D space.

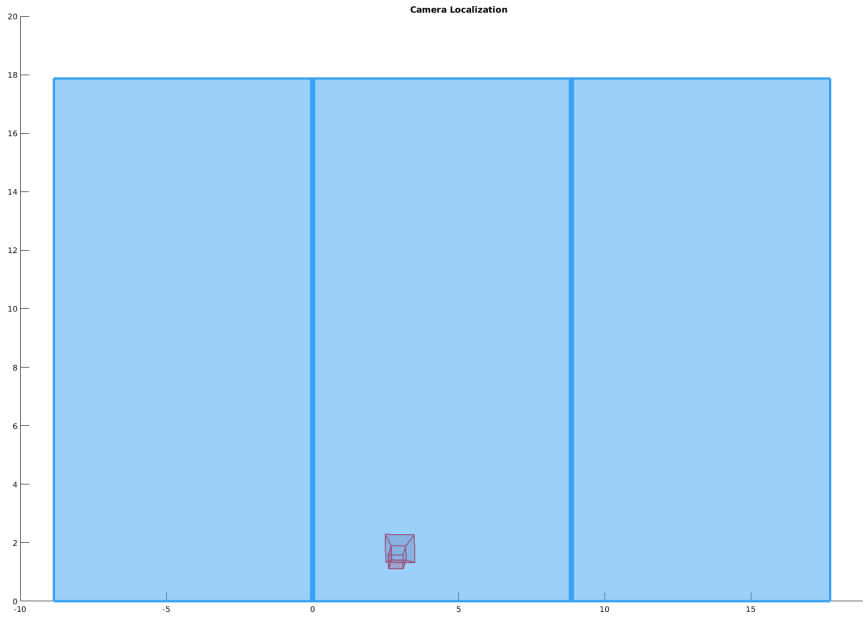
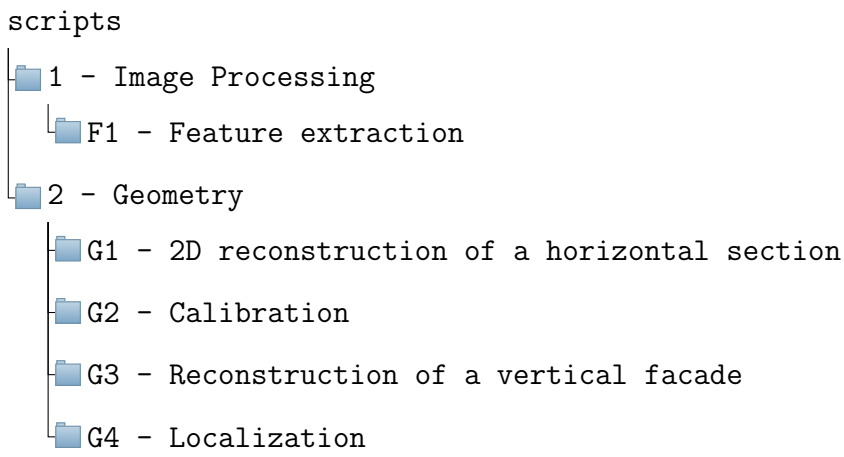


Figure 17: Front Scene in the 3D space.

The 3D building has been reconstructed by exploiting the ratios computed in the previous points.

Repository Instructions

The Matlab repository is structured into two main folders:



In the folder "`scripts/1 - Image Processing/F1 - Feature extraction`" there are:

- `main_edges.m`
- `main_corners.m`
- `main_lines.m`

While in each folder contained in "`scripts/2 - Geometry/*/`" there is always the file:

- `main.m`

With respect to the related request contained in the folder.
Note. In some files executions it is necessary to press the *Enter* button in order to have a sequential execution (paused in advance for the sake of simplicity).

References

Hartley, R. I. and Zisserman, A. – "*Multiple View Geometry in Computer Vision*", 2nd ed. – Cambridge University, 2004.