Image Analysis and Computer Vision Homework

Giorgio Romeo

Politecnico di Milano
A.Y. 2021/2022

January 25, 2022

POLITECNICO
MILANO 1863

Contents

1 Introduction

The homework is based on the back side of Villa Melzi d'Eril in Bellagio. In the bottom view below (Fig. 1), all lines lie on a same horizontal plane Π. In addition the facades 1 and 5 are coplanar, and both are parallel to facade 3. Furthermore, facades 2 and 4 are perpendicular to facades 1, 3 and 5. The sun is placed at the point at the infinity $S = [3.9 \quad -1 \quad z \quad 0]^\intercal$ -where z is irrelevant- with respect to the reference frame reported in the picture. The layout of the architectonic elements on facade 3 is symmetric with respect to a central vertical axis. Windows of facades 1, 3 and 5 are are equally wide. Windows of facades 2 and 4 can not be exploited in reconstruction since they are poorly visible.

An image of the back side of Villa Melzi d'Eril is taken by a digital camera (Fig. 2). The camera skew factor is assumed to be null; the aspect ratio is unknown (thus natural camera can not be assumed), as well as the principal point and the focal distance. The camera height (z-coordinate) over the ground plane is 1.5 m.

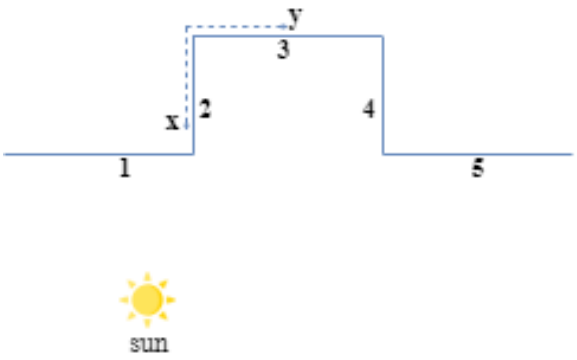


Figure 1: Scene

Figure 2: Back side of Villa Melzi d'Eril

2 Feature extraction

Combining the learned techniques, find edges, corner features and straight lines in the image.
The features were extracted in three steps:

- Edge detection using Canny algorithm
- Line detection using Hough transform
- Corner detection using Harris-Stevens algorithm

2.1 Edge detection

The Canny algorithm relies on a multi-stage edge detector. It can be divided in three steps:

- *Computation of the magnitude and angle of the directional gradients*: it uses a filter based on the derivative of a Gaussian in order to compute the intensity of the gradients. The Gaussian reduces the effect of noise present in the image.
- *Non-Maximum suppression*: the image magnitude produces results in thick edges but the final image should have thin edges. Thus, to thin out the edges, we perform non-maximum suppression by finding the pixel with the maximum value in an edge
- *Hysteresis thresholding*: some edges may not actually be edges and there is some noise in the image. Double thresholding deals with this problem. A pixel is considered an *edge pixel* if its gradient is above "high" threshold, a *non-edge pixel* if its gradient is below "low" threshold. Furthermore, if the gradient at a pixel is between "low" and "high" thresholds then declare it an edge pixel if and only if it can be directly connected to an edge pixel or connected via pixels between "low" and "high".

The threshold parameters found for Canny algorithm are the result of many experiments. Note that the line extraction part is affected by this final result. The processed image with extracted edges is shown in figure 3.

Figure 3: Extracted edges of the image

2.2 Line detection

The Hough transform is a technique that can be used to isolate features of a particular shape within an image. In our case, the Hough transform was used to detect straight lines. Since the Cartesian representation for lines does not include vertical lines, it is more convenient to utilize the parametric notation: $rho = x * cos(\theta) + y * sin(\theta)$.

Each datum (edge point extracted using Canny algorithm) votes for all models compatible with it. The followed steps are:

- Discretization of the Hough Space into cells
- For each cell, set a vote counter equals to 0
- For each datum:
  - Compute the Hough Transform
  - Determine cells crossed by Hough Transform
  - Increment vote counter for crossed cells
- Select cells where vote counter is greater than a threshold and is a local max
- Refine line estimate by refitting to points that voted that line

The result of the application of the Hough Transform for line detection is shown in Fig. 4.

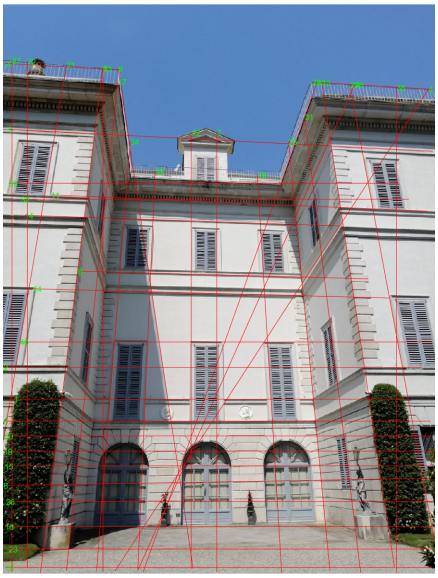


Figure 4: Extracted lines of the image

2.3 Corner detection

A corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Harris-Stevens algorithm considers a small window (in our case 3 x 3) around each pixel in the image. We want to identify all pixel windows that are unique, where uniqueness can be measured by shifting each window by a small amount in a given direction and measuring the amount of change that occurs in the pixel values.

We compute the sum squared difference (SSD) of the pixel values before and after the shift and identify pixel windows where the SSD is large for shifts in all 8 directions. We define the change function E(u,v) as the sum of all the SSD, where u,v are the x,y coordinates of every pixel in our 3 x 3 window and I is the intensity value of the pixel. The features in the image are all pixels that have large values of E(u,v) with respect to a pre-defined threshold. The algorithm identifies a corner if the SSD is be large in shifts for all eight directions.

The processed image with extracted corners is shown in figure 5.

Figure 5: Extracted corners of the image

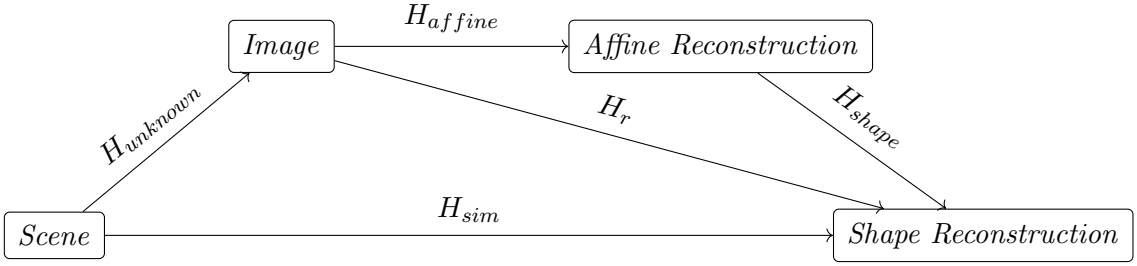
3 2D reconstruction of a horizontal section

Rectify (2D reconstruct) the horizontal section of the building from the useful selected image lines and features, including vertical shadows. In particular, determine the ratio between the width of facade 2 (or 4) and the width of facade 3.

We performed 2D reconstruction of the horizontal section of the building utilizing a stratified approach instead of a direct one in order to increase robustness (numerical errors should decrease). The stratified approach consists of performing sequentially an affine reconstruction and a shape reconstruction of the original image. We first computed the affine mapping H_{affine} that maps the original image to its affine reconstruction. Then, we computed the similarity mapping H_{shape} that maps the obtained affine reconstruction to its shape reconstruction. The overall mapping H_r that maps the original image to its shape reconstruction can be computed as follows:

$$H_r = H_{shape} * H_{affine}$$

The overall stratified approach can be summarized in the following diagram:



3.1 Affine reconstruction

An affine transformation is any transformation preserving: parallelism, ratio of areas, ratio of length on collinear or parallel lines, linear combination of vectors, line at the infinity, invariants of a projective transformation. To compute H_{affine} we exploited the fact that a projective mapping H maps the line at the infinity $l_\infty = [0 \quad 0 \quad 1]^\intercal$ onto itself if and only if H is affine. Thus, this mapping must map any point x' belonging to l'_∞ (image of the l_∞) onto a point at the infinity, namely:

$$H_{affine}\, x' = \begin{bmatrix} * \\ * \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} * & * & * \\ * & * & * \\ & l'^\intercal_\infty & \end{bmatrix} x' = \begin{bmatrix} * & * & * \\ * & * & * \\ & l'^\intercal_\infty\, x' & \end{bmatrix} = \begin{bmatrix} * \\ * \\ 0 \end{bmatrix} \tag{1}$$

Taking into account that H_{affine} should have the previous form and be non-singular, we chose the following matrix representation (the simplest one):

$$H_{affine} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ & l'^{\mathsf{T}}_{\infty} & \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -0.00000795 & -0.00082198 & 1 \end{bmatrix} \tag{2}$$

For the computation of the line at the infinity l'_{∞}, we exploited the blue points highlighted in Fig. 6 (the green points were utilized in the shape reconstruction). First of all, we considered an horizontal plane parallel to the floor highlighting the blue points a, b, c and d. Afterwards, we computed the first vanishing point $vp1$ (not represented in the figure) as the intersection of the two lines ab and dc (images of parallel lines in the real scene), while the second vanishing point $vp2$ as the intersection of the two lines ad and bc (images of parallel lines in the real scene). Eventually, we computed l'_{∞} as the cross product of the two vanishing points (they belong to the image of the line at the infinity), obtaining the mapping H_{affine} as previously stated. Note that "line ab" should be properly defined as "line passing through point a and b" (similar for all other lines).

The resulting affine reconstruction obtained applying H_{affine} to the original image is shown in Fig. 7. Note that the affine transformation H_{affine} was also applied to the blue (a, b, c, d) and green (g) points to represent them in the affine reconstruction.

Figure 6: Features and lines in the original image used for affine and shape reconstruction

Figure 7: Affine reconstruction of the original image

3.2 Shape reconstruction

A similarity transformation is any transformation preserving: ratio of length, angles, circular points, invariants of both projective and affine transformations. To compute H_{shape} we exploited the fact that a projective mapping H maps the circular points $\mathbf{I}$ and $\mathbf{J}$ onto themselves if and only if H is a similarity. A crucial property is that, after an affine reconstruction, the dual conic $C_\infty^{*'}$ image of the (original) conic dual to the circular points $(\mathbf{I}, \mathbf{J})$ can be written as:

$$C_\infty^{*'} = \begin{bmatrix} KK^\intercal & 0 \\ 0^\intercal & 0 \end{bmatrix} \tag{3}$$

where K is a 2 x 2 invertible matrix. $S = KK^\intercal$ is a symmetric and homogeneous matrix, thus there are only 2 unknowns to identify $C_\infty^{*'}$. We found the first unknown exploiting known angles: given two lines $l = [l_1 \quad l_2 \quad l_3]^\intercal$ and $m = [m_1 \quad m_2 \quad m_3]^\intercal$, the term $l_1m_1 + l_2m_2$ can be written as:

$$\begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = l^\intercal C_\infty^* m \tag{4}$$

Furthermore, from the transformation rules we can derive: $l^\intercal C_\infty^* m = l'^\intercal C_\infty^{*'} m'$. Thus, we may compute the angle θ between two lines as:

$$\cos(\theta) = \frac{l_1m_1 + l_2m_2}{\sqrt{(l_1^2 + l_2^2)(m_1^2 + m_2^2)}} = \frac{l^\intercal C_\infty^* m}{\sqrt{(l^\intercal C_\infty^* l)(m^\intercal C_\infty^* m)}} = \frac{l'^\intercal C_\infty^{*'} m'}{\sqrt{(l'^\intercal C_\infty^{*'} l')(m'^\intercal C_\infty^{*'} m')}} \tag{5}$$

Referring to Fig. 7, we selected the pair of lines ad and dc (images of orthogonal lines in the real scene) to exploit the fact that $\cos(\theta) = 0$ for orthogonal lines. Thus, equation (5) can be simplified as follows:

$$\cos(\theta) = 0 = \frac{l'^\intercal C_\infty^{*'} m'}{\sqrt{(l'^\intercal C_\infty^{*'} l')(m'^\intercal C_\infty^{*'} m')}} \Rightarrow l'^\intercal C_\infty^{*'} m' = 0 \tag{6}$$

More specifically, we wrote equation (6) as:

$$\begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix} * \begin{bmatrix} a & b & 0 \\ b & c & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = l_1m_1a + l_2m_1b + l_1m_2b + m_2l_2c = 0 \tag{7}$$

Eventually, we obtained the first constraint:

$$\begin{bmatrix} l_1m_1 & l_1m_2 + l_2m_1 & l_2m_2 \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{8}$$

Note that we could not select another pair of orthogonal lines such as dc and cb for the second constraint because this one would not have been independent from the previous constraints. For the last constraint, we exploited the invariance of ratio of length in similarity transformations.

First of all, as shown in Fig. 8, exploiting the fact that the sun is a point at the infinite along the direction [3.9 -1] (z is irrelevant) and triangles 1 and 2 are similar (same angles), we computed the ratio k between the lengths of segments gd and ad: $k = 1/3.9$. Segment ad represents the width of facade 2, while gd represents the width of the shadow segment defined from point d (junction of facade 2 and 3) and point g (intersection of the vertical shadow eg and the plane defined by the blue points (a, b, c, d), as shown in Fig. 6). The two segments were computed as:

$$gd = \begin{bmatrix} g'_1 - d'_1 \\ g'_2 - d'_2 \\ g'_3 - d'_3 \end{bmatrix} = \begin{bmatrix} l'_1 \\ l'_2 \\ l'_3 \end{bmatrix} = l', \quad ad = \begin{bmatrix} a'_1 - d'_1 \\ a'_2 - d'_2 \\ a'_3 - d'_3 \end{bmatrix} = \begin{bmatrix} m'_1 \\ m'_2 \\ m'_3 \end{bmatrix} = m' \tag{9}$$

Furthermore, exploiting the fact that $l/m = 1/3.9 = k$, we computed:

$$l_1^{'2} + l_2^{'2} = k^2(m_1^{'2} + m_2^{'2}) \Rightarrow l'^{\intercal} C_\infty^{*'} l' = k^2(m'^{\intercal} C_\infty^{*'} m') \tag{10}$$

More specifically, we wrote equation (10) as:

$$\begin{bmatrix} l_1^{'2} & 2l'_1 l'_2 & l_2^{'2} \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix} = k^2 \begin{bmatrix} m_1^{'2} & 2m'_1 m'_2 & m_2^{'2} \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{11}$$

Eventually, we obtained the second constraint:

$$\begin{bmatrix} l_1^{'2} - k^2 m_1^{'2} & 2(l'_1 l'_2 - k^2 m'_1 m'_2) & l_2^{'2} - k^2 m_2^{'2} \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{12}$$

Once we found the image of the dual conic $C_\infty^{*'}$, using the single value decomposition (SVD) method, we computed:

$$SVD(C_\infty^{*'}) = UDV^{\intercal} = H_{shape}^{-1} C_\infty^* H_{shape}^{-1\intercal} \tag{13}$$

However, since we did not obtain:

$$D = C_\infty^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

we exploited the fact that:

$$D = \begin{bmatrix} \sqrt{D_{11}} & 0 & 0 \\ 0 & \sqrt{D_{22}} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{D_{11}} & 0 & 0 \\ 0 & \sqrt{D_{22}} & 0 \\ 0 & 0 & 0 \end{bmatrix} = D_S C_\infty^* D_S \tag{15}$$

obtaining:

$$UDV^\intercal = UD_S C_\infty^* D_S V^\intercal \tag{16}$$

Eventually, we computed:

$$H_{shape} = (UD_S)^{-1} = \begin{bmatrix} 1.17870261 & -0.24436278 & 0 \\ -0.24436278 & 1.42341806 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

The resulting shape reconstruction obtained applying H_{shape} to the affine reconstruction is shown in Fig. 9. Note that the shape transformation H_{shape} was also applied to the blue (a, b, c, d) and green (g) points to represent them in the shape reconstruction.

Eventually, we computed the overall mapping H_r as:

$$H_r = H_{shape} * H_{affine} = \begin{bmatrix} 1.17870261 & -0.24436278 & 0 \\ -0.24436278 & 1.42341806 & 0 \\ -0.00000795 & -0.00082198 & 1 \end{bmatrix} \tag{18}$$

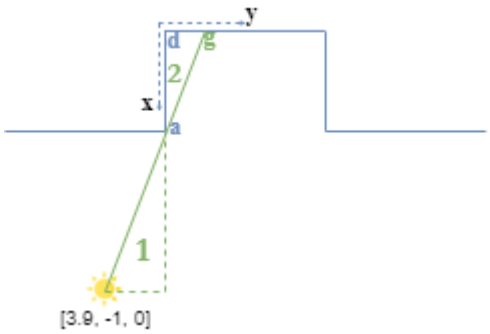


Figure 8: Similar triangles

Figure 9: shape reconstruction of the horizontal section of the buiding

3.3 Ratio results

The final task of this point was to compute the ratio between the width of facade 3 and width of facade 3. Thus, referring to Fig. 9, we measured the length of segment ad (width

facade 2) and length of segment dc (width facade 3), obtaining:

$$\frac{Facade\_2_{width}}{Facade\_3_{width}} = 0.659764$$

Even if not required, we also computed the ratio between the width of facade 3 and width of the shadow segment dg, both after affine and shape reconstruction. We obtained the same result (invariance of the ratio of length):

$$\frac{Facade\_3_{width}}{Shadow\_Segment_{width}} = 6.891124$$

4 Calibration

First extract a vertical vanishing point and then use it together with useful information extracted during the rectification step, in order to estimate the calibration matrix K containing the intrinsic parameters of the camera, namely: focal distance, aspect ratio and position of principal point.
Calibration matrix K is defined as:

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The relation between the image of the absolute conic ω and the calibration matrix K is:

$$\omega = (KK^t)^{-1} \tag{20}$$

Furthermore, for a zero skew camera ω is given by:

$$\omega = \begin{bmatrix} \alpha^2 & 0 & -u_0\alpha^2 \\ 0 & 1 & -v_0 \\ -u_0\alpha^2 & -v_0 & f_y^2 + \alpha^2 u_0^2 + v_0^2 \end{bmatrix} \tag{21}$$

that is a symmetric matrix with 4 degrees of freedom, hence just 4 constraints on ω were required. For computing the constraints we needed:

- the overall mapping $H_r = H_{shape} * H_{affine}$
- the image of the line at the infinity l'_∞ (see section 3.1)
- the vanishing point $vp_{vertical}$ along the direction orthogonal to the horizontal plane defined by points (a, b, c, d) (see Fig. 6). In particular, referring to Fig. 10, we computed the vertical vanishing point as the intersection between lines he and gf (images of parallel lines in the real scene).

The constraints were computed exploiting:

- the image of the circular point $\mathbf{I}'$:

$$\mathbf{I}' = H_r^{-1}\mathbf{I} = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} = h_1 + ih_2 \quad \text{and} \quad (h_1 + ih_2)^\intercal \omega (h_1 + ih_2) = 0 \tag{22}$$

  leading to constraints:

$$h_1^\intercal \omega h_2 = 0 \quad \text{and} \quad h_1^\intercal \omega h_1 = h_2^\intercal \omega h_2 \tag{23}$$

- the line at infinity on the horizontal plane is orthogonal with respect to the vanishing point of the vertical direction:

$$[l'_\infty]_\times \; \omega \; vp_{vertical} = 0 \tag{24}$$

  where $[l'_\infty]_\times$ is the vector product matrix of l'_∞.

- the orthogonality of vanishing points belonging to orthogonal directions (such as the ones computed in section 3.1):

$$vp_1 \; \omega \; vp_2 = 0 \tag{25}$$

Figure 10: Lines to compute the vertical vanishing point

Once we obtained the image of the absolute conic ω, we computed the calibration matrix K exploiting relation (21):

$$K = \begin{bmatrix} 1273.58016001317 & 0 & 540.535823296631 \\ 0 & 975.253444381331 & 875.511631430140 \\ 0 & 0 & 1 \end{bmatrix} \tag{26}$$

where

$$f_x = 1273.58016001317$$
$$f_y = 975.253444381331$$
$$u_0 = 540.535823296631$$
$$v_0 = 875.511631430140$$

5 Reconstruction of a vertical facade

Use the knowledge of K to rectify also a vertical facade, as, e.g., facade 3. Knowing the calibration matrix K we computed the image of the absolute conic as:

$$\omega = (KK^t)^{-1} \tag{27}$$

Referring to Fig. 11, we computed the first vertical (along the vertical direction) vanishing point $vp3$ ($vp_{vertical}$ in equation (24)) as the intersection between lines he and gf (images of parallel lines in the real scene) and the second vertical vanishing point $vp4$ as the intersection between lines hg and ef (images of parallel lines in the real scene).

Since the absolute conic contains all the circular points, i.e. the circular points I_π, J_π of any plane belong to the absolute conic, the image ω of the absolute conic contains the image I'_π, J'_π of all the circular points I_π, J_π. Thus, we computed the image of the circular points of the vertical plane (facade 3) as the intersection between the image of the absolute conic ω and the image of the line at the infinity to which vertical vanishing points vp3 and vp4 belong. Afterwards, we computed the dual conic $C^{*'}_\infty$ image of the (original) conic dual to the circular points $(\mathbf{I}, \mathbf{J})$ as:

$$C^{*'}_\infty = \mathbf{I'}\mathbf{J'}^\intercal + \mathbf{J'}\mathbf{I'}^\intercal \tag{28}$$

Once we found the image of the dual conic $C^{*'}_\infty$, using the single value decomposition (SVD) method, we computed:

$$SVD(C^{*'}_\infty) = UDV^\intercal = H^{-1}_{rect} C^*_\infty H^{-1\intercal}_{rect} \tag{29}$$

As what we did in section 3.2, we exploited the fact that:

$$D = \begin{bmatrix} \sqrt{D_{11}} & 0 & 0 \\ 0 & \sqrt{D_{22}} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{D_{11}} & 0 & 0 \\ 0 & \sqrt{D_{22}} & 0 \\ 0 & 0 & 0 \end{bmatrix} = D_S C^*_\infty D_S \tag{30}$$

obtaining:

$$UDV^\intercal = UD_S C^*_\infty D_S V^\intercal \tag{31}$$

Eventually, we computed the rectification homography as:

$$H_{rect} = (UD_S)^{-1} \tag{32}$$

The result of the reconstruction of vertical facade 3 are shown in Fig. 12, where we also represented the green points highlighted in the original image in Fig. 11.

Figure 11: Lines used to compute the two vertical vanishing points

Figure 12: Reconstruction of facade 3

6 Localization

Determine the relative pose (i.e. position and orientation) between facade 3 and the camera reference. Use information about the camera height to solve for scale.

Finding the relative pose between facade 3 (highlighted through points (e, f, g, h) in Fig. 12) and the camera reference was possible knowing the size and shape of facade 3, the image and the calibration matrix K.

Identifying the plane of the vertical facade 3 with π and the position of a point in the plane reference frame with X_π, we can write the position of a point in the world reference frame as:

$$X_w = [R_\pi | o_\pi] X_\pi \tag{33}$$

Referring to Fig. 12, the plane reference frame has the y axis identical to the line at the bottom of facade 3, going right, the z axis with the same direction of the left line he and the x axis orthogonal to both axis using the right hand rule. Thus, the origin of the plane reference frame is represented by point h. In this way a point on the plane can be written

(in homogeneous coordinates) as:

$$X_\pi = \begin{bmatrix} 0 \\ y \\ z \\ w \end{bmatrix} \tag{34}$$

Defining as P the projection matrix:

$$P = [KR| - KRo] \tag{35}$$

mapping 3D points (X) of the world to points in the image (x). From:

$$x = PX \tag{36}$$

we obtained:

$$u = [KR| - KRo][R_\pi|o_\pi]X_\pi \tag{37}$$

By putting the world reference frame on the camera ($R = I$ and $o = [0\ 0\ 0]^\intercal$):

$$u = [K|\underline{\mathbf{0}}] \begin{bmatrix} j_\pi & k_\pi & o_\pi \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y \\ z \\ w \end{bmatrix} \Rightarrow u = K[j_\pi|k_\pi|o_\pi]x \tag{38}$$

Where x are the coordinates of the point on the plane. By inspection we identified $K[j_\pi|k_\pi|o_\pi] = H_{homography}$ the matrix that maps the real points to the image. H was known since we knew shape and size of facade 3 (transformation mapping the shape of facade 3 face to the image). So we obtained:

$$[j_\pi|k_\pi|o_\pi] = K^{-1}H_{homography} \tag{39}$$

Where $H_{homography} = [h_1 \mid h_2 \mid h_3]$ is the transformation mapping world points to image points. The matrix $R = [i_\pi \mid j_\pi \mid k_\pi]$ (rotation of the plane with respect to the camera) can be found using these equations:

$$\begin{aligned} \lambda &= \frac{1}{|K^{-1}h_1|} \\ j_\pi &= K^{-1}h_1\lambda \\ k_\pi &= K^{-1}h_2\lambda \\ i_\pi &= j_\pi \times k_\pi \\ o_\pi &= K^{-1}h_3\lambda \end{aligned}$$

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

$$SVD(R) = [U,\ \sim\ ,\ V] \Rightarrow \hat{R} = UV \tag{41}$$

The computed camera position and camera rotation are:

$$cameraPosition = \begin{bmatrix} 17.96874075 \\ 2.41349869 \\ 1.5 \end{bmatrix}$$

$$cameraRotation = \begin{bmatrix} 0.10673388 & -0.32533008 & -0.93955745 \\ 0.99411473 & 0.01729609 & 0.10694268 \\ -0.01854100 & -0.94544232 & 0.32526151 \end{bmatrix}$$

The results of the localization are shown in the following figures:

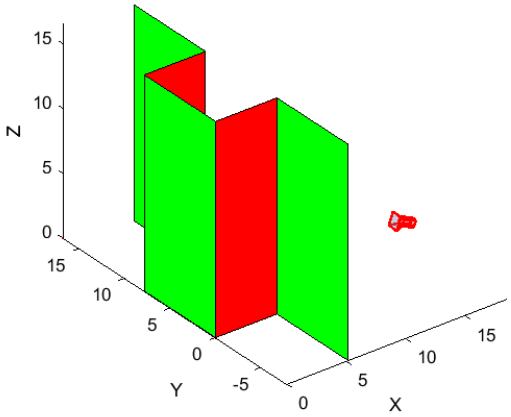


Figure 13: Camera location (1)

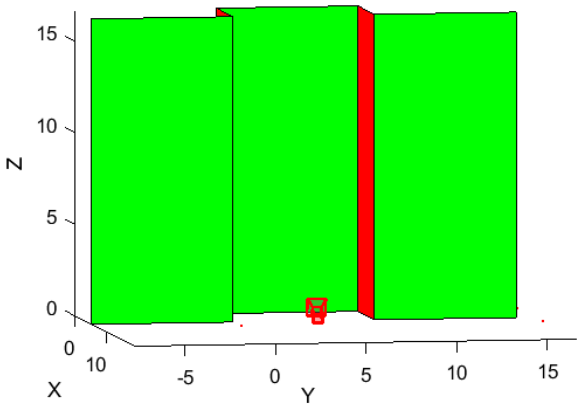


Figure 14: Camera location (2)

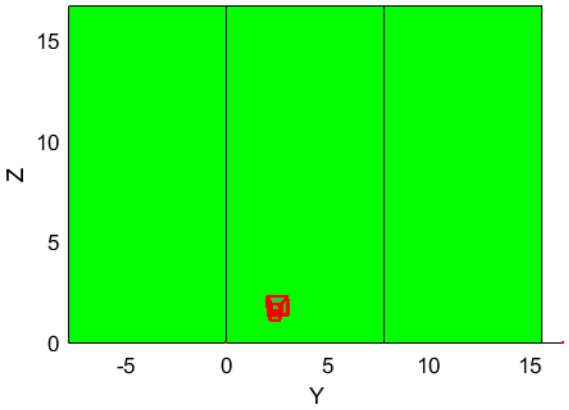


Figure 15: Camera location along plane YZ

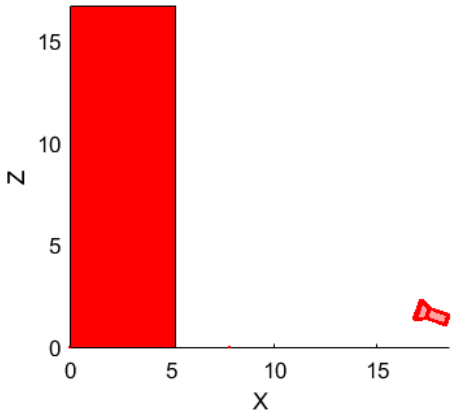


Figure 16: Camera location along plane XZ

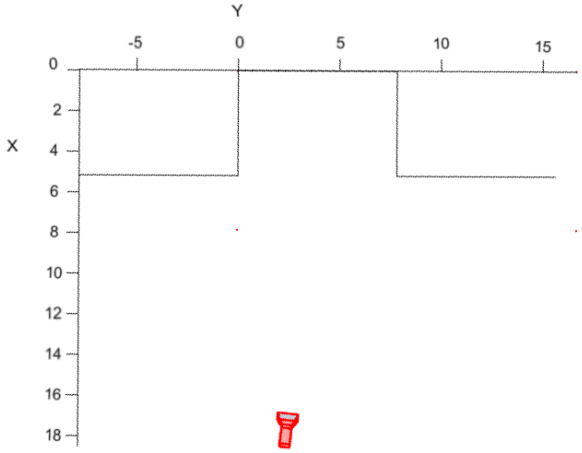


Figure 17: Camera location along plane XY (1)

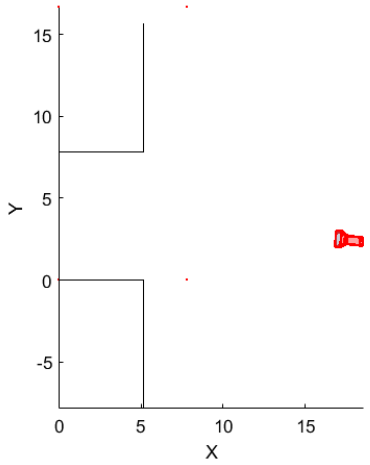


Figure 18: Camera location along plane XY (2)

7 References

- A Flexible new Technique for Camera Calibration (Zhang)
- Multiple View Geometry in Computer Vision (Hartley, Zisserman)
- Image Analysis and Computer Vision course slides Politecnico di Milano a.y. 2021/2022