Image Analysis and Computer Vision Homework

Giorgio Romeo

Politecnico di Milano
A.Y. 2021/2022

January 22, 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 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.

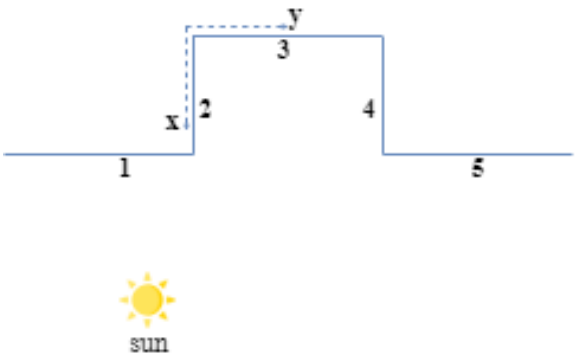


Figure 1: Scene

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 is 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 steps followed 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.

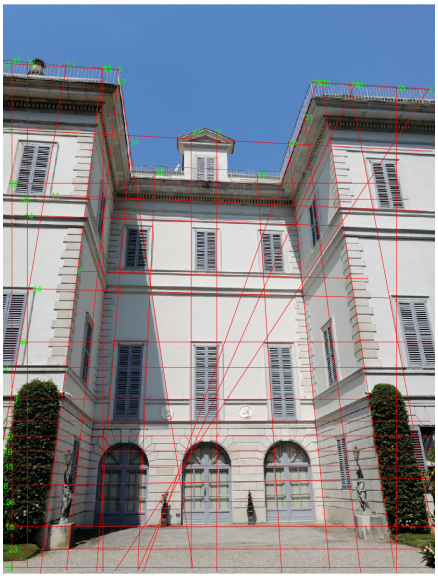


Figure 4: Extracted lines of the image

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

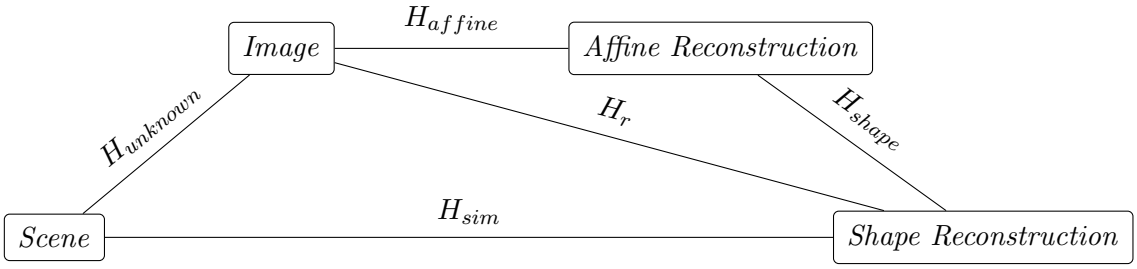
3 2D reconstruction of a horizontal section

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):

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

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 point 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.

The resulting affine reconstruction obtained applying H_{affine} to the original image is shown in Fig. 7.

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 out 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 $l_1 m_1 + l_2 m_2$ 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:

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

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. Indeed, 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:

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

Eventually, we obtained the first constraint:

$$\begin{bmatrix} l_1 m_1 & l_1 m_2 + l_2 m_1 & l_2 m_2 \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{8}$$